Contribution à la modélisation dynamique des systèmes articulés. Bases mathématiques et outils informatiques

Ali Hamlili

THESE DE DOCTORAT

présentée

à

L'ECOLE NATIONALE DES PONTS ET CHAUSSEES

Spécialité

MATHEMATIQUES ET INFORMATIQUE

CONTRIBUTION A LA MODELISATION DYNAMIQUE DES SYSTEMES ARTICULES

(BASES MATHEMATIQUES ET OUTILS INFORMATIQUES)

---

par

ALI HAMLILI

Soutenue le 17 Septembre 1993, devant le Jury d'Examen composé de:

Mme	PASCAL Madeleine	Président Rapporteur
MM.	LAZARD Daniel	Rapporteur
	CHEVALLIER Dominique	Directeur de thèse
	LALEMENT René	Examinateur
	LERBET Jean	Examinateur
	RIGOLOT Alain	Examinateur

ABSTRACT

During the last years, mechanical articulated systems have become increasingly important in the field of automation. This work makes two important contributions towards a better utilization of mathematical abstraction:

- The first contribution concerns the dynamic modelisation of articulated systems like mechanical manipulators and complex open kinematic chains. The mathematical abstraction by the Lie groups and Lie algebras theory offers an excellent means for simplifying the syntactical form of expressions of models. New methods for describing (by fundamental families) configurations of mechanical systems and an original efficient recursive computational scheme of Newton-Euler dynamics are developed. With this formulation, it should be possible to compute a near-optimal Newton-Euler dynamics in real time.

- The second contribution concerns algebraic typing, term rewriting theory and automatic generation of codes. These problems lead to new computer algebra system architectures based on artificial intelligence methods and representations in multi-equational specifications. In this order of ideas, a prototype of computer algebra system (*SURVEYOR*) and an extension (*MEDUSA MF77*) of Maple system, based on object oriented programming and artificial intelligence control, are realized. A software tool for generating Fortran and Maple iterative symbolic optimized codes of our dynamic formulation is developed with *MEDUSA MF77* system.

Keywords:

Kinematics, Dynamics, Simple and Complex Open Kinematic Chains, Differential Geometry, Lie Groups and Lie Algebras, Artificial Intelligence, Object Oriented Programming, Automatic Generation of Codes.

REMERCIEMENTS

Il m'est agréable d'exprimer dans ces quelques lignes ma reconnaissance et ma gratitude aux personnes qui ont contribué à la réussite de ce travail:

Monsieur *Dominique CHEVALLIER*, Directeur-adjoint du CERMA, a dirigé cette thèse. Il m'a prodigué conseils constructifs et remarques pertinentes tout au long de ces années de recherche à ses cotés. Qu'il me permette de lui témoigner ma profonde reconnaissance.

Madame *Madeleine PASCAL*, Professeur de l'université d'Evry Val d'Essonne, a accepté d'être président rapporteur du jury. Je lui exprime tout mon respect et ma gratitude.

Monsieur *Daniel LAZARD*, Professeur de l'Université Pierre et Marie Curie, a accepté d'être rapporteur de ce travail. Je le remercie vivement pour l'intérêt qu'il a manifesté à l'égard de mes travaux et pour l'enthousiasme avec lequel il a lu ce document.

Monsieur *René LALEMENT*, Professeur à l'ENPC et Directeur-adjoint du CERMICS, je le remercie sincèrement pour ses remarques valeureuses concernant la partie informatique de ce travail.

Monsieur *Jean LERBET*, Maître de conférence à l'université de Tours, acceptera que j'évoque ici l'aide amicale qu'il m'a apporté dans la hâte des derniers jours. Je l'en remercie cordialement.

Monsieur *Alain RIGOLOT*, Professeur des universités, m'a fait l'honneur d'examiner mes travaux et de participer au jury de thèse. Je le remercie pour l'enthousiasme chaleureux avec lequel il m'a fait part de ses remarques.

J'adresse mes sincères remerciements à la direction de l'école nationale des ponts et chaussés et aux membres du centre d'enseignement et de recherche en mathématiques appliquées qui m'ont accueilli aux seins de leur centre pour toute la période de ma thèse. Ils ont su écouter mes exposés, leurs remarques et leurs questions m'ont été précieuses. Je n'oublie pas de remercier Madame *Véronique SERRE* secrétaire du CERMA pour sa gentillesse et ses compétences multiples, ainsi qu'à tous mes camarades du centre pour leur amitié.

Enfin, je dédie ce travail à mes parents et ma famille. Je leur serai éternellement reconnaissant pour leur soutien et leur amour.

TABLE DES MATIERES

LISTE DES FIGURES

INTRODUCTION

0.1 Contexte de notre travail

L'un des objectifs de la recherche en mécanique est l'analyse mathématique du comportement des systèmes articulés (massivement utilisés par la technologie moderne), d'expliquer ce comportement par des modélisations et ce afin de faire des prévisions ou de le reproduire par la commande. L'entrelacement entre différentes disciplines des sciences appliquées et la difficulté croissante des systèmes font de l'informatique, coordonnée avec l'étude mathématique des modèles, un outil indispensable à cette analyse.

Ainsi, la complexité croissante des systèmes mécaniques, le progrès des automates programmables et la variété des domaines des applications mécaniques des systèmes articulés (industries manufacturières, spatiales, sous-marines, nucléaires, transports, ... etc) exigent une adaptation permanente de la formalisation mathématiques et la conception de nouveaux modèles et algorithmes performants pour le calcul du comportement de tels systèmes. Cette coopération des mathématiques et de l'informatique a ouvert un important domaine de recherche où la conception et la réalisation des systèmes d'aide à la modélisation et à la génération des schémas de calcul assistées par ordinateur sont des outils indispensables. De nombreux systèmes informatiques ont été consacrés à une partie ou une autre de ces questions [R. Sturges[STUR-73], W. Khalil [KHAL-78], J.Y.S. Luh [LUH—81], M.G. Aldon [ALDO-82], J. Wittenburg & U. Woltz [WITT-85], Ch. Garnier [GARN-90]].

Les multiples faces des problèmes que nous venons de soulever peuvent être classées en deux catégories:

- Les problèmes de modélisation d'ordre mathématique: Un modèle est une représentation simplifiée de la réalité. Il repose sur un ensemble d'hypothèses simplificatrices décrivant une partie du modèle réel et de son fonctionnement. Pour tous les systèmes articulés, qu'ils soient motorisés ou non, il existe plusieurs modes de fonctionnement (ou de commande): par

apprentissage, en position, en vitesse et dynamique. L'intérêt de la description basée sur le modèle dynamique est de permettre notamment:

- de prendre en compte les termes:
  - d'inertie, des forces centrifuges et de Coriolis,
  - éventuellement, de frottement, de déformation, ... etc.
- d'évaluer les forces et couples transitoires des actionneurs:
  - pour dimensionner les efforts,
  - pour déterminer au mieux les rapports des transmetteurs.

Les objectifs technologiques et industriels qui motivent le choix d'un tel modèle sont essentiellement liés à la précision et la répétabilité de la tâche d'une part et l'accélération du rythme de travail et de production d'autre part. En effet, une commande dynamique permet notamment d'accroître la vitesse de travail en gardant:

- la même précision que pour la commande en position,
- la même répétabilité qu'en commande par apprentissage.

Cependant, les difficultés dues à la complexité du modèle dynamique résident essentiellement dans le volume du calcul matriciel et les incidences qu'il peut induire sur le facteur temps réel lorsque cette complexité n'est pas maîtrisées.

- Les problèmes d'ordre informatique liés à la conception et la réalisation d'un système d'aide à la modélisation et à la génération de codes répondant aux "facteurs de qualité". Ces facteurs de qualité doivent recouvrir deux aspects:
  - Les qualités générales du logiciel d'aide à la modélisation: convivialité, ergonomie, rapidité, fiabilité, modularité, souplesse, adaptabilité, ... etc.
  - Les qualités spécifiques liées à la cohérence, la modularité et la compréhensibilité du code source généré par le système logiciel:
    - décomposition modulaire: le système logiciel doit réduire la complexité apparente des équations générées en mettant en facteur les calculs communs afin d'optimiser le code source,
    - précédence modulaire: les sous-modules du code source doivent être disposés selon l'ordre de précédence des calculs; c'est à dire, de telle manière qu'à l'exécution des programmes, les variables figurant dans le sous-module en cours de traitement doivent être évaluées à des étapes antérieures,
    - compréhension modulaire: d'une part, les sous-modules doivent avoir un sens (dans le cas qui nous intéresse, ils doivent avoir un sens mécanique) de façon à ce que

l'utilisateur puisse comprendre les étapes du calcul. D'autre part, le système logiciel doit être capable de déclarer automatiquement dans les "zones de déclaration" du code source: le type, la dimension et la précision de calcul des différentes variables. Le cas échéant, le système doit pouvoir commenter automatiquement les sous-modules du code source pour aider les utilisateur à mieux comprendre le déroulement du processus de calcul des programmes.

Les solutions, proposées par la résolution informatique, à ces problèmes relèvent de deux catégories:

- La première catégorie, dite algorithmique, correspond aux cas où l'on dispose de modèles mathématiques (c'est le cas du problème de calcul du modèle dynamique). La solution consiste alors à choisir les mathématiques comme langage de programmation: pour cela, on dispose entre autres d'un large choix de systèmes de calcul formel permettant une bonne représentation du contexte mathématique du problème.
- La seconde catégorie de solutions correspond aux cas où l'on dispose de connaissances et d'heuristiques générales[1] (c'est le cas des problèmes d'obtention des facteurs de qualités du code généré). L'intelligence artificielle propose, dans de tels cas, des systèmes basés sur des règles de production (i.e. des modules de la forme: *condition* $\Longrightarrow$ *action*), des règles de réécriture, des stratégies de résolution, des heuristiques, ... etc. Cette démarche ambitieuse cherche à combiner la simulation du raisonnement mental humain aux performances de rapidité de l'ordinateur.

L'intérêt d'un approche hybride (Algorithmique–I.A.) dans le problème qui nous préoccupe vient du fait que des algorithmes, des concepts, des axiomes, des lois, ... et même des jugements de bon sens ou intuitifs coopéreront ensemble pour atteindre la solution finale. C'est le type de solution que nous préconisons dans cette thèse.

0.2 Fondements et buts de la thèse

Le but principal de cette thèse est donc le développement des outils nécessaires à l'élaboration d'un algorithme performant pour le calcul du modèle dynamique en temps réel, sa généralisation à une large catégorie des systèmes articulés -arborescents- et enfin la conception et la réalisation des outils informatiques basés sur des systèmes de calcul formel, pour la génération automatique:

- d'un schéma de calcul symbolique itératif optimisé en nombre d'opérations élémentaires du modèle dynamique dans le langage Fortran;

[1] Notons que dans nombre de cas, ces connaissances et heuristiques peuvent se ramener à des descriptions par des modèles mathématiques. Mais, pas dans le sens de la modélisation mathématique invoquée au paragraphe précédent. Dans ce cas, il s'agit surtout de problèmes de gestion d'hypothèses, de gestion arborescente, de planification, ... etc. On peut donc recourir selon les cas à divers types de représentations: bases de règles, logique, probabilités, recherche opérationnelle, ... etc.

- des équations formelles explicites du modèle dynamique dans le langage du calcul formel.

Dans cet esprit, la partie mathématique de notre thèse poursuit les travaux de D. Chevallier sur la modélisation des systèmes articulés à l'aide de la théorie des groupes et algèbres de Lie [CHEV-86]. L'auteur a exposé, dans cet article, les aspects algébriques du groupe $\mathbb{D}$ des déplacements et la structure de variété différentielle de l'espace $\mathcal{S}$ des configurations d'un solide rigide. L'existence d'un produit intérieur "[.|.]", qui munit $\mathcal{S}$ d'une structure rimannienne invariante par $\mathbb{D}$, permet d'exposer la théorie newtonnienne de l'inertie d'un solide rigide et d'aboutir à certaines propriétés particulières de l'opérateur d'inertie. Une description axiomatique des liaisons et la relation entre vitesses des corps en liaison permettent d'aboutir à une formulation analytique des équations explicites de la dynamique à partir du principe des puissances virtuelles.

Nous nous somme inspirés également des travaux de J-M. Hervé [HERV-78] sur la représentation formelle des liaisons dans les paires cinématiques par des sous-groupes de déplacements. Mais, notre approche est différente dans la mesure où elle se base sur la caractérisation de tels sous-groupes par les générateurs infinitésimaux de leurs décompositions relatives à des *familles fondamentales*[2]. En effet, afin d'atteindre la rigueur du raisonnement analytique de la représentation formelle, nous avons cru nécessaire de faire ressortir l'aspect infinitésimal de cette modélisation. Nous nous sommes attaché dans les différents points traités dans ce mémoire à mettre de l'ordre dans les idées pratiques et de la rigueur mathématique dans les fondements théoriques. Nous avons d'ailleurs obtenu, indépendamment des travaux précédents et conformement aux ambitions exprimés au début de cette introduction, nombre de résultats nouveaux. Nous avons cherché à exprimer un formalisme ou n'interviennent que des structures (algébriques) et des opérations (propices aux traitement informatique); pour cela il était nécessaire de substituer à la notion de repères affines orthonormés directs la notion intrinsèque de familles fondamentales.

0.3 Etat de la question de dynamique

Aujourd'hui, toutes les méthodes de génération des équations du mouvement résultent de l'une des deux approches de base, à savoir:

- **Première approche**: il s'agit de la traduction du mouvement par un ensemble d'équations algébro-différentielles déduites des théorèmes généraux de la mécanique classique. Dans cette classe se retrouvent toutes les formulations qui sont basées sur le formalisme de Newton-Euler.

- **Seconde approche**: dans cette approche, on exprime les équations de la dynamique d'un système articulé holonome à l'aide d'un système différentiel du second ordre sur la variété différentielle représentant l'espace des configurations. Les équations sont généralement

[2] C'est une notion que nous proposons comme alternative à la notion classique de repères affines orthonormés directs et dont nous exposerons l'intérêt et les avantages plus loin dans ce texte.

exprimées dans des systèmes de coordonnées et/ou quasi-coordonnées et peuvent revêtir diverses formes classiques: équations de Lagrange, équations de Gibbs–Appel, formalisme de Hamilton, formalisme de Kane; traduisant ainsi une des multiples formes du principe des puissances virtuelles.

La recherche contemporaine dans ce domaine concerne deux grandes catégories de problèmes de dynamique:

(a) **Modèle dynamique inverse** : ou tout simplement modèle dynamique; est le problème qui consiste à chercher les efforts (i.e. forces/couples) qu'il faut appliquer aux actionneurs pour produire ou reproduire un mouvement caractérisé par la donnée des positions, vitesses et accélérations articulaires du système articulé. Les méthodes exposées dans la littérature revêtent deux formes:

- La forme explicite où le vecteur des efforts articulaires sont déterminés par des équation de la forme:

$$Q = A(q)\,\ddot{q} + b(q,\dot{q})$$

où:

Q	vecteur d'ordre N (où N est le nombre de degrés de liberté de la structure articulée) des couples/forces des actionneurs, selon la nature des articulations (rotoïde ou prismatique),
q	vecteur des N positions articulaires,
$\dot{q}$	vecteur des vitesses articulaires,
$\ddot{q}$	vecteur des accélérations articulaires,
$A(q)$	matrice d'ordre $N \times N$, généralement appelée matrice de l'énergie cinétique ou matrice d'inertie du robot du fait que l'énergie cinétique s'écrit $E = \frac{1}{2}\,\dot{q}^t\,A(q)\,\dot{q}$,
$b(q,\dot{q})$	vecteur des termes de Coriolis et des forces centrifuges, non-linéaires en fonction de q et $\dot{q}$, et des termes des forces de gravité.

- Forme récursive (implicite) où l'on ne peut séparer les différentes variables articulaires, mais qui présente de nombreux avantages comme nous le montrerons dans cette thèse.

(b) **Modèle dynamique directe** : dans ce problème, il s'agit d'exprimer les accélérations articulaires en fonction des positions, vitesses et efforts des actionneurs.

Ainsi, plusieurs formalismes ont été utilisés pour obtenir les équations du mouvement des systèmes mécaniques articulés [E.T. Whitteker [WHIT-04], W. Hooker & G. Margulies [HOOK-65], V.I. Arnold [ARNO-66], R. Roberson & J. Wittenburg [ROBE-66], J. Wittenburg [WITT-77], J.Y.S. Luh, M.W. Walker & R.P.C. Paul [LUH—80], T.R. Kane & D.A. Levinson [KANE-80], J.M. Hollerbach [HOLL-80], M. Renaud [RENA-80], P. Coiffet [COIF-81], D. Chevallier & Helmer [CHEV-84], M. Vukobratović [VUKO-85], D. Chevallier [CHEV-86], M. Valášek [VALA-91]].

0.3.1 Formalisme de Lagrange

Le formalisme des équations de Lagrange consiste à exprimer des équations différentielles du second ordre à l'aide de l'*énergie cinétique*. La formulation standard du modèle dynamique basée sur ce formalisme pour des systèmes articulés à structure quelconque découle des travaux de J.J. Uicker [UICK-65]. Cette méthode générale est basée sur le calcul de la trace de matrices et représente ainsi un puissant outil de calcul. Mais, son défaut majeur est qu'elle présente trop de calculs redondants. Plusieurs auteurs ont utilisé cette méthode [Kahn [KAHN-69], Kahn & B. Roth [KAHN-71], J.M. Haollerbach [HOLL-80], M. Renaud [RENA-80]].

0.3.2 Formalisme de Gibbs–Appel

Les travaux de bases qui sont à l'origine de ce formalisme remontent à la fin du XIX^e siècle [J.W. Gibbs [GIBB-879], P. Appell [APPE-00] & [APPE-11]]. Les équations sont exprimées à l'aide de l'*énergie d'accélération* et de la *dérivation d'Appell*. S'appuyant sur ce principe, A.F. Vereschagin [VERE-74] a proposé un modèle dynamique explicite pour les mécanismes à structure de chaîne dont les articulations sont de type prismatique ou pivot. Plus récemment, T.R. Kane et D.A. Levinson [T.R. Kane & D.A. Levinson [KANE-80], [KANE-83] & [KANE-85]] ont formulé sur ces bases ce qu'on appelle *les équations de Kane*, utilisées depuis par plusieurs auteurs [E.A. Desloge [DESL-87], J.E. Keat [KEAT-87], D.E. Rosenthal [ROSE-87]].

0.3.3 Formalisme de Newton–Euler

Ce formalisme découle des théorèmes généraux de la mécanique. Il est fréquemment utilisé dans la commande dynamique des systèmes articulés. L'étude faite par Hooker et G. Margulies [HOOK-65] présente l'une des premières tentatives de systématisation du modèle dynamique pour des systèmes mécaniques à structures arborescentes. Cette formulation a été améliorée par R. Roberson et J. Wittenburg en introduisant la théorie des graphes [R. Roberson & J. Wittenburg [ROBE-66]]. Cependant, l'inconvénient de ces méthodes est qu'il est souvent difficile de prévoir l'élimination des efforts intérieurs à la structure articulée. P.W. Likins a apporté une solution à ce problème en ramenant toutes les articulations à un seul degré de liberté donnant forme ainsi à la première méthode explicite de la formation du modèle dynamique [P.W. Likins [LIKI-71] & [LIKI-74]]. C'est dans cette même période, que les équations de Newton–Euler ont été utilisées pour la première fois dans la modélisation des structures biomécanique telles que le corps humain [R.L. Huston & C. E. Passerello [HUST-71], Stepanenko & Vukobratović [STEP-76], Orin & al. [ORIN-79]].

En 1979, J.Y.S. Luh, M.W. Walker et R.P.C. Paul ont proposé une méthode itérative très adaptée à la commande dynamique en temps réel des robots à chaîne cinématique simple [LUH—80]. Depuis, cette méthode a été utilisée sous diverses formes par de nombreux auteurs [J.M. Hollerbach [HOLL-80] , E. Dombre & W. Khalil [DOMB-88]]. Par ailleurs, plusieurs auteurs ont développé des

méthodes d'analyse topologique pour les formulations récursives de la dynamique des mécanismes [J. Wittenburg [WITT-77], R.S. Hwang & E.J. Haug [HWAN-89]].

En 1966, V.I. Arnold a publié un article [ARNO-66] consacré à l'étude du mouvement d'Euler–Poinsot par la théorie des groupes et algèbres de Lie où il exprime notamment les équations d'Euler à partir de l'étude des géodésiques du groupe classique $SO(3)$. Sur les mêmes bases théoriques, dans les années 80, D. Chevallier a proposé une méthode analytique explicite permettant de formuler les équations de la dynamique de façon intrinsèque dans l'algèbre de Lie associée au groupe des déplacements à partir du principe des puissances virtuels pour des systèmes multicorps avec des articulations quelconques [D. Chevallier [CHEV-86]].

En utilisant la notion des nombres duaux, découverte par le mathématicien W.K. Clifford [CLI-873] & [CLI-878] et utilisée depuis en cinématique par de nombreux auteurs [A.T. Yang [YAN-64a], A.T. Yang and F. Freudenstein [YAN-64b], F.M. Dimentberg [DIME-65], G.R. Veldkamp [VELD-82], D. Chevallier [CHEV-91]], nous avons développé une formulation itérative du modèle dynamique des robots manipulateurs [A. Hamlili [HAML-91]]. En 1992, nous avons amélioré cette formulation [HAML-92] pour donner forme à un algorithme basé sur une double récurrence, d'une efficacité quasi-optimale, très adapté au calcul du modèle dynamique des structures articulés en temps réel.

0.4 Plan et apports de la thèse

Ainsi, comme on peut le voir après ces quelques lignes, le but pratique de notre travail dans le domaine de la dynamique ne peut être atteint qu'au prix de développements théoriques et trouve sa confirmation par comparaison avec les résultats classiques de la mécanique des systèmes articulés de corps rigides. Ce mémoire de thèse comporte quatre parties.

0.4.1 La partie 1

Il est souvent intéressant et parfois nécessaire de changer la représentation d'un problème afin de se ramener à un problème pour lequel on dispose d'outils d'analyse plus naturels permettant d'exprimer les idées très clairement et avec moins d'efforts. La première partie à pour objet de définir les bases théoriques préliminaires aux théories abordées pour établir nos modèles. Elle comporte des développements et outils mathématiques fondamentaux. Ainsi, elle est essentielle à la compréhension des raisons qui ont motivé nos choix autant dans la modélisation géométrique que dynamique du comportement des systèmes articulés.

Le chapitre 1 s'attache aux éléments de la théorie des nombres duaux et les méthodes de base permettant leur implémentation en calcul symbolique. Les raisons qui nous ont amenés à étudier les extensions sur l'anneau des nombres duaux et à dégager nombre de résultats qui semblent ne

pas être étudiés ailleurs sont au nombre de deux:

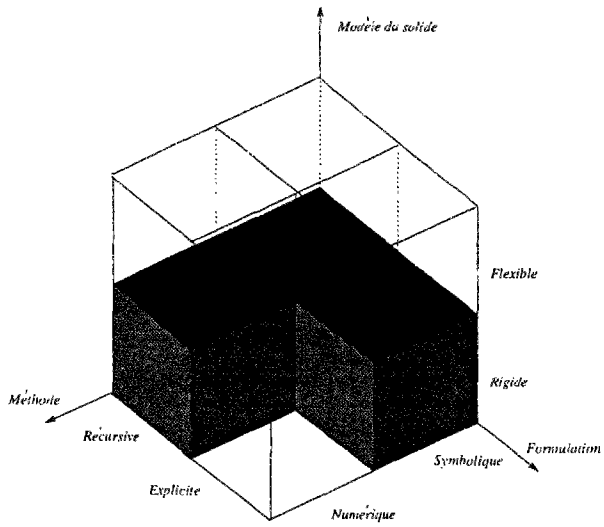
- la première raison et la plus importante se rattache à l'implantation de telles structures dans des processus de calcul formel;
- la seconde raison est l'absence, dans la littérature, d'un travail rigoureux qui puisse servir de référence de base sur ce point particulier.

Le chapitre 2 est consacré à la présentation de l'ensemble des outils mathématiques que nous allons adopter pour la modélisation des aspects géométriques et dynamiques des systèmes articulés dans le cadre de la théorie des groupes et algèbre de Lie. Dans ce chapitre, nous proposons les notions des familles fondamentales et familles mobiles comme une alternative aux notions classiques de repères affines orthonormés directs et repères mobiles de l'espace géométrique affine euclidien. En substance, ces deux notions ne traduisent aucune autre prétention que celle "d'algébriser" totalement la notion de repère affine orthonormé direct. Ainsi, elle apporte une clarification formelle des calculs (cette façon de voire les choses est étroitement liée à la représentation des configurations d'un solide par celles d'un trièdre orthonormé direct). Les idées que nous proposerons, seront méthodiquement développées par des séries de théorèmes et de propositions donnant ainsi un corpus mathématique à ces aspects théoriques de la modélisation.

0.4.2 La partie 2

La deuxième partie se compose également de deux chapitres. Elle présente les modèles de comportement des systèmes articulés à structure de chaîne cinématique ouverte simple ou arborescente dans le contexte théorique défini dans la première partie.

Le chapitre 3 est consacré à l'étude du modèle géométrique des systèmes articulés à chaînes cinématiques ouvertes simples et arborescentes. Les aspects développés dans le chapitre 2 nous ont conduit naturellement à proposer la généralisation des définitions des angles d'Euler et des angles de Briant permettant ainsi la description des déplacements et des mouvements de 1 à 6 degrés de liberté. Grâce à l'outil de calcul formel et compte tenu des résultats démontrés au chapitre 1 sur l'inversibilité des prolongements canoniques duaux, nous exprimerons les conditions d'application de telles conventions généralisées. L'identification des paramètres de description du mouvement va permettre, entre autres, de déterminer le modèle géométrique inverse et les configurations singulières pour de tels paramétrages du groupe des déplacements. Enfin, ce chapitre se termine en établissant des modèles géométriques itératifs pour les mécanismes à structure de chaîne cinématique ouverte simple ou arborescente. La traduction des conventions de Denavit–Hartenberg dans ce langage mathématique permettra de mettre en évidence un algorithme itératif simple pour exprimer une méthode analytipue pour déterminer tels paramètres à partir de la donnée des familles fondamentales associées aux élément de la chaîne cinématique.



Cette figure schématise les champs de la recherche en dynamique des systèmes multicorps suivant trois axes cardinaux. La région colorée situe les champs de d'action de cette thèse dans le vaste domaine de la recherche en dynamique des systèmes articulés.

Fig. 0.1: Champs de recherche concernés par notre travail

Le chapitre 4 porte entièrement sur la modélisation dynamique des mécanismes à structure de chaîne cinématique ouverte simple ou arborescente. Nous définissons notamment dans ce chapitre la notion de pseudo-accélération, le procédé de calcul d'un tel champ de vecteurs et sa relation avec les notions classiques d'accélérations angulaire et linéaire en un point du solide. L'expression, dans ce formalisme, des lois fondamentales de la dynamique et du bilan des efforts agissant sur chaque corps d'un système articulé permettent alors de déduire un algorithme performant pour le calcul du modèle dynamique (applicable au calcul en temps réel). Nous proposerons alors une comparaison, basée sur le coût de calcul, de cet algorithme avec sept autres algorithmes de référence. Enfin, ce chapitre se termine par la généralisation de cet algorithme au cas des systèmes articulés arborescents. Dans ce contexte, les modèles que nous allons proposer présenteront un intérêt double: simplicité syntaxique des modèles d'une part et description de la réalité dans des cas complexes d'autre part.

0.4.3 La partie 3

Dans cette troisième partie, nous considérons les problèmes relatifs à la génération d'un code optimisé du modèle dynamique itératif. Nous présenterons le prototype *SUREVEYOR* et le logiciel *MEDUSA MF77* que nous avons développés à cet effet. Sur un exemple simple, nous montrons que les fonctionnalités actuelles des systèmes de calcul formel classiques ne permettent pas encore la production de codes qui requièrent les facteurs souhaités de qualité. Nous exposerons ensuite les solutions que nous apportons à ces problèmes par la réécriture dans des algèbres libres (abstractions algébriques) de termes du premier ordre (i.e. contenant des variables) au moyen des techniques de l'intelligence artificielle.

Enfin, cette partie se termine par l'exposé, sur des exemples, des principales fonctions du module `dualstruc.map` de calcul formel sur les objets duaux. Ce module définit de puissants outils informatiques de calcul et de manipulation formelle des objets duaux pour le mathématicien, d'autant plus certains d'entre eux sont même spécialiser en mécanique. Le but de leur conception est de permettre d'écrire et manipuler des expressions mathématiques formelles à partir d'objets duaux. L'ensemble de ces outils permet d'exprimer dans une syntaxe simple des modèles de réalités matérielles et physiques très complexes.

0.4.4 Les annexes

La première annexe propose un rappel succinct des différents résultats de la théorie générale des groupes et algèbres de Lie. Elle pourrait ainsi servir de référence rapide aux lecteurs. Les autres annexes proposent des applications variées des différentes méthodes proposées dans cette thèse à des problèmes d'ingénierie.

Partie 1:

ASPECTS MATHEMATIQUES

– *Résumé de la partie 1* –

PRELIMINAIRES MATHEMATIQUES ET BASES DE LA MODELISATION

Il est souvent intéressant et parfois nécessaire de changer la représentation d'un problème afin de se ramener à un problème pour lequel on dispose d'outils d'analyse plus naturels qui permettent d'exprimer les idées très clairement et avec moins d'efforts. Dans cette première partie, nous nous fixons pour objet de définir les bases théoriques et les fondements mathématiques préliminaires aux thèmes abordés pour établir nos modèles. Ainsi, elle essentielle à la compréhension des raisons qui ont motivé nos choix autant sur le plan de la modélisation géométrique que dynamique du comportement des systèmes articulés.

Enfin, nous verrons comment ces méthodes exactes peuvent servir pour l'algébrisation de notions physiques dont l'utilisation sous la forme classique, dans des développements formels, est souvent responsable à la complication des calculs et des modèles. Nous montrerons, plus loin dans ce mémoire, que l'ensemble des moyens que nous proposons font de nos méthodes des outils indispensables à une détermination systématique de modèles généraux des systèmes articulés. Les modèles s'exprimeront alors dans ce langage mathématique sous des formes syntaxiques relativement simples.

Chapitre 1

ELEMENTS DE LA REPRESENTATION DUALE

1.1 Introduction

Le mathématicien William Kingdon Clifford (1845–1879) a introduit une idée très originale sur la théorie des "moteurs"[1] qui réside dans l'utilisation d'un certain opérateur ε qui permet d'exprimer symboliquement un "moteur" (M, M_0) sous forme d'un vecteur spécial[2]: $M + \varepsilon M_0$ où ε est un opérateur nilpotent du second ordre.

C'est ainsi, à la suite des travaux de W.K. Clifford [[CLI-873] & [CLI-878]], que l'existence d'un tel opérateur a été implicitement acceptée. Plus tard, en se basant sur les travaux de Ball et Zanchevskiy sur la théorie des vis et de ceux de Kotelńikov sur l'application des nombres duaux à la théorie des vecteurs, F.M. Dimentberg a proposé une large application du calcul dual sur les vis pour le traitement des problèmes de cinématique [F.M. Dimentberg [DIME-65]]. Dans de récents travaux, plusieurs auteurs [K. Sugimoto & J. Duffy [SUG-82a], G.R. Veldkamp [VELD-82], A.K. Pradeep, P.J. Yoder & R. Mukundan [PRAD-89], D. Chevallier [CHEV-91]] ont proposé des présentations modernes des applications de tels nombres en cinématique.

Dans le chapitre présent, nous allons introduire la notion de nombres duaux comme un outil mathématique. Nous nous attacherons à mettre de la rigueur mathématique dans notre exposé afin de ressortir les idées réelles qui existent derrière l'utilisation des nombres duaux. Avec cet outil, nous allons construire un certain nombre de structures algébriques adjacentes et établir nombre

[1] Nous avons utilisé ici la traduction du mot anglais: Motor.

[2] L'introduction de ce type de vecteurs qui résulte d'une telle construction formelle a conduit Clliford à des résultats intéressants [DIME-65]:

- Les opérations sur les moteurs sont indépendantes du point de l'espace géométrique où ils sont réduits.
- La partie vectorielle d'un moteur apparaît comme le résultat de l'opération par ε du même moteur.

de résultats analytiques et algébriques afin de mettre en évidence les propriétés caractéristiques qui nous permettrons d'envisager un calcul symbolique direct (et systématique) sur cette notion. Contrairement à ce que l'on pourrait croire, travailler sur des structures algébriques et des outils basés sur les nombres duaux n'est pas une simple généralisation[3] du calcul sur ces mêmes notions dans $\boldsymbol{R}$. Il s'agit en fait de travailler dans de nouvelles structures algébriques qui n'ont plus les mêmes propriétés que les structures algébriques correspondantes sur $\boldsymbol{R}$. En effet, même si les axiomes d'un module sont identiques à ceux d'un espace vectoriel, au changement près du corps de base en un anneau, et même si les principales définitions d'algèbre linéaire sont valables dans les modules, les propriétés algébriques d'un module peuvent différer substantiellement de celles d'un espace vectoriel. En particulier, des propriétés intuitives de la géométrie ou de l'existence d'une norme peuvent être en défaut dans un module. Et manque d'une justification mathématique rigoureuse des résultats, on ne peut affirmer avec certitude la validité des modèles généralisés.

1.2 Nombres duaux

Dans ce paragraphe nous allons définir les éléments de base du calcul sur les nombres duaux. Les éléments de ce langage sont indispensables pour aborder des structures algébriques duales plus complexes.

1.2.1 Définition structurelle

En un certain sens, les nombres duaux peuvent être conçus comme des doublets[4] de nombres réels munis de certaines lois de composition interne.

Définition 1.1

On appellera ensemble des nombres duaux et on va noter Δ, l'ensemble $\boldsymbol{R}^2$ muni des lois d'addition et de multiplication, sur des couples de nombres réels, définies respectivement par:

$$(x_1, x_2) + (y_1, y_2) = \quad (x_1 + y_1, x_2 + y_2) \qquad \text{(Eq.1.1)}$$

et

$$(x_1, x_2)\,(y_1, y_2) = \quad (x_1\, y_1, x_1\, y_2 + x_2\, y_1) \qquad \text{(Eq.1.2)}$$

Dès à présent un nombre de remarques immédiates peuvent être faites sur la structure algébrique de Δ:

- D'abord, l'ensemble Δ a une structure d'anneau unitaire commutatif pour les lois d'addition et de multiplication définies ci-dessus.
- Son élément unité est: $1_\Delta = (1, 0)$.

[3] Cependant, il est vrai que la restriction des résultats qu'on peut établir sur les Δ-modules, aux structures réelles correspondent aux résultats classiques sur les $\boldsymbol{R}$-espaces vectoriels à des isomorphismes près.

[4] D'autres définitions peuvent être envisagés. A titre d'exemple, l'ensemble des nombre duaux peut être considéré comme étant une algèbre quadratique de type (0,0) [N. Bourbaki [BOUR-70]], ou encore la structure $\boldsymbol{R}[X]/[X^2]$.

- De plus l'élément $\varepsilon = (0,1)$ est nilpotent d'ordre deux; c'est à dire:

$$\varepsilon^2 = 0_\Delta \qquad \text{(Eq.1.3)}$$

L'ensemble Δ n'est donc pas un anneau intègre, puisqu'il contient des diviseurs de zéro.

- Par ailleurs, l'ensemble $\{(a,0)/\, a \in \boldsymbol{R}\}$ est un sous-anneau de Δ isomorphe à $\boldsymbol{R}$. Ainsi, dans la suite, nous pourrons identifier cet ensemble à $\boldsymbol{R}$.

- L'ensemble Δ peut être muni d'une structure d'espace vectoriel de dimension deux sur $\boldsymbol{R}$, dont une base canonique est donnée par la famille $\{1_\Delta, \varepsilon\}$, de sorte que l'on peut désormais identifier Δ à $\boldsymbol{R} \oplus \varepsilon\, \boldsymbol{R}$. On s'accordera donc d'écrire, de façon symbolique, pour tout nombre dual $x = (x_1, x_2)$:

$$x = x_1 1_\Delta + \varepsilon\, x_2 \equiv x_1 + \varepsilon\, x_2\,; \quad \text{avec } x_1, x_2 \in \boldsymbol{R} \qquad \text{(Eq.1.4)}$$

1.2.2 Parties réelle et partie duale d'un nombre dual

Définition 1.2

Nous appellerons parties réelle et duale d'un nombre dual, les fonctions Re et Du à arguments dans Δ et à valeurs dans $\boldsymbol{R}$ qui à tout nombre dual $x = x_1 + \varepsilon\, x_2$ de l'ensemble Δ, associent respectivement les composantes réelles:

$$Re(x) = x_1 \;\; et \;\; Du(x) = x_2$$

On a l'habitude d'utiliser un support géométrique pour représenter intuitivement la notion de produit cartésien. Cependant, bien que cet approche soit assez commode pour se faire une idée intuitive, il ne faut pas trop s'y fier pour les démonstrations mathématiques. Ainsi, il peut arriver que l'on rencontre dans la littérature le terme de points du plan Δ. A chaque point sont associées deux coordonnées et, de ce point de vue, les parties réelle et duale d'un nombre dual sont respectivement les projections sur les axes de la première et la seconde coordonnée du point du plan qui représente ce nombre.

1.2.3 Nombres duaux inversibles – Quotient

D'après la relation (Eq.1.2) qui définit la multiplication sur les nombres duaux, les éléments inversibles de Δ sont les nombres duaux $x = x_1 + \varepsilon\, x_2$, avec $x_1 \neq 0$, tels que:

$$x^{-1} = \frac{1}{x_1} - \varepsilon \frac{x_2}{{x_1}^2} \qquad \text{(Eq.1.5)}$$

Corollaire 1.1

Soient deux nombres duaux x et y, avec $Re(y) \neq 0$. Si $\dfrac{x}{y}$ désigne le quotient de x par y, alors:

$$Re\left(\frac{x}{y}\right) = \frac{Re(x)}{Re(y)} \;\; et \;\; Du\left(\frac{x}{y}\right) = \frac{Re(y)\,Du(x) - Re(x)\,Du(y)}{Re(y)^2} \qquad \text{(Eq.1.6)}$$

1.3 Fonction d'une variable duale

1.3.1 Définition – Exemples

Définir une fonction f, d'une variable duale $x = x_1 + \varepsilon x_2$, revient à définir une application d'un domaine $D \subseteq \boldsymbol{R}^2$ dans $\boldsymbol{R}^2$, telle que: $f(x) = f_1(x_1, x_2) + \varepsilon f_2(x_1, x_2)$, où f_1 et f_2 peuvent être considérées comme deux fonctions de deux variables réelles et à valeurs dans $\boldsymbol{R}$.

Exemples

A titre d'exemples, on peut définir des fonctions sinus et cosinus duales, pour tout nombre dual $x = x_1 + \varepsilon x_2$, par:

$$\widetilde{sin}\, x = \sin x_1 + \varepsilon x_2 \cos x_1 \text{ et } \widetilde{cos}\, x = \cos x_1 - \varepsilon x_2 \sin x_1 \qquad \text{(Eq.1.7)}$$

Si aucune confusion n'est à craindre, on peut noter tout simplement *sin* et *cos* au lieu de $\widetilde{sin}$ et $\widetilde{cos}$, ces fonctions sinus et cosinus duales.

Lemme 1.2

Soit n un nombre entier naturel quelconque. Alors, pour tout nombre dual x:

$$x^n = Re(x)^n + \varepsilon\, n\, Du(x)\, Re(x)^{n-1} \qquad \text{(Eq.1.8)}$$

Preuve :

Δ étant un anneau commutatif, on peut appliquer la formule du binôme en tenant compte de $\varepsilon^2 = 0$. Ainsi, si $x = x_1 + \varepsilon x_2$, alors:

$$x^n = \sum_{k=0}^{n} C_n^k\, {x_1}^{n-k}\, \varepsilon^k\, {x_2}^k = {x_1}^n + \varepsilon\, n\, x_2\, {x_1}^{n-1}$$

Ce qu'il fallait démontrer. ∎

Proposition 1.3

Soit x un nombre dual arbitraire, alors, les séries formelles suivantes sont convergentes dans Δ *et on a:*

$$\sum_{n=1}^{\infty} \frac{(-1)^{n-1}\, x^{2n}}{(2n)!} = 1 - \cos x \qquad \text{(Eq.1.9)}$$

et

$$\sum_{n=0}^{\infty} \frac{(-1)^n\, x^{2n+1}}{(2n+1)!} = \sin x \qquad \text{(Eq.1.10)}$$

Preuve :

La démonstration est immédiate, elle découle du lemme 1.2, de la définition des fonctions sinus et cosinus duales et du développement en série des fonctions sinus et cosinus de variable réelle. ∎

Plus généralement, toutes les identités établies pour la trigonométrie réelle, peuvent être étendues, quand elles sont définies, à la trigonométrie duale en tenant compte des propriétés d'inversibilité[5] des nombres duaux.

[5] Je fait allusion ici aux formules où pourrait intervenir un quotient dual.

1.3.2 Limite et continuité de fonctions de variable duale

Considérons maintenant $x = x_1 + \varepsilon\, x_2$ et $a = a_1 + \varepsilon\, a_2$. Nous dirons q'une fonction de variable duale $f(x) = f_1(x_1, x_2) + \varepsilon f_2(x_1, x_2)$ tend vers une limite $b = b_1 + \varepsilon\, b_2$ lorsque $x \to a$ et nous noterons $\lim\limits_{x \to a} f(x) = b$, si $f(x)$ tend vers b au sens de la topologie de $\boldsymbol{R}^2$. C'est à dire aussi:

$$\lim_{x \to a} f(x) = b \iff \begin{cases} \lim\limits_{\substack{x_1 \to a_1 \\ x_2 \to a_2}} f_1(x_1, x_2) = b_1 \\ \\ \lim\limits_{\substack{x_1 \to a_1 \\ x_2 \to a_2}} f_2(x_1, x_2) = b_2 \end{cases} \qquad \text{(Eq.1.11)}$$

Ainsi, nous dirons que la fonction de variable duale est continue en un point a, si et seulement si elle est continue en (a_1, a_2) pour la topologie de $\boldsymbol{R}^2$:

$$\lim_{x \to a} f(x) = f(a) \qquad \text{(Eq.1.12)}$$

et elle sera dite continue sur un domaine $D \subseteq \boldsymbol{R}^2$, si elle est définie et continue en tout point de D.

1.3.3 Δ–différentiabilité – Dérivée d'une fonction de variable duale

Par définition, une fonction f de variable duale sera dite Δ–différentiable, en un point x, lorsqu'il existe un nombre dual $A \in \Delta$ et une fonction de variable duale Θ tels que :

$$f(x + h) = f(x) + A\, h + h\, \Theta(h) \text{ et } \lim_{\substack{h \in \Delta \\ h \to 0}} \Theta(h) = 0 \qquad \text{(Eq.1.13)}$$

Ainsi, la valeur A sera dite: dérivée de la fonction de variable duale f au point x. Plus généralement, une fonction f de variable duale sera dite Δ–différentiable sur un domaine D ($\subset \Delta$) lorsqu'elle est Δ–différentiable en tout point de D. Dans ces conditions, la fonction $f' : x \mapsto A(x)$, ainsi définie, sera dite fonction dérivée de la fonction f.

1.3.4 Conditions de Δ–différentiabilité

Nous voulons déterminer dans ce paragraphe une condition nécessaire et suffisante pour qu'une fonction de variable duale soit Δ–différentiable.

Théorème 1.4

Soit une fonction de variable duale f, notons f_1 et f_2 les fonctions, de $\boldsymbol{R}^2$ dans $\boldsymbol{R}$, qui définissent ses parties réelle et duale. Alors:

$$f \text{ est } \Delta\text{–différentiable} \iff \begin{cases} f \text{ est } \boldsymbol{R}^2\text{–différentiable} \\ \\ \dfrac{\partial f_1}{\partial x_1} = \dfrac{\partial f_2}{\partial x_2} \\ \\ \dfrac{\partial f_1}{\partial x_2} = 0 \end{cases} \qquad \text{(Eq.1.14)}$$

Preuve :

Considérons une fonction de variable duale f définie par:

$$f(x) = f_1(x_1, x_2) + \varepsilon\, f_2(x_1, x_2) \text{ où } f_1, f_2 \in \mathcal{F}(\boldsymbol{R}^2, \boldsymbol{R}) \text{ et } x = x_1 + \varepsilon\, x_2 \in \Delta$$

(a) Si f est Δ–différentiable, alors d'après la relation (Eq.1.13), l'application:

$$\begin{array}{ccc} \boldsymbol{R}^2 & \to & \boldsymbol{R}^2 \\ h & \mapsto & A\,h \end{array}$$

est une application linéaire -particulière- de $\boldsymbol{R}^2$ dans $\boldsymbol{R}^2$. En effet, posons $A = a + \varepsilon\, b$, $h = h_1 + \varepsilon\, h_2$ et écrivons le produit matriciel dans $\boldsymbol{R}^2$ associé au produit $A\,h$ dans Δ:

$$h' = [h'_1\ h'_2]^t = [a\,h_1 \quad a\,h_2 + b\,h_1]^t = \begin{pmatrix} a & 0 \\ b & a \end{pmatrix} [h_1\ h_2]^t$$

or h' n'est autre que l'image de h par la matrice jacobienne de f dans $\boldsymbol{R}^2$ et ceci pour tout $h \in \boldsymbol{R}^2$. Donc:

$$\begin{pmatrix} a & 0 \\ b & a \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix}$$

Ainsi, on conclut que:

$$\frac{\partial f_1}{\partial x_1} = \frac{\partial f_2}{\partial x_2} \text{ et } \frac{\partial f_1}{\partial x_2} = 0$$

(b) La réciproque est triviale.

Ce qu'il fallait démontrer. ∎

Corollaire 1.5

Si $D = I \times \boldsymbol{R}$, où I est un ouvert de $\boldsymbol{R}$, et si f est une fonction de variable duale Δ–différentiable sur D. Alors:

(a) $x_2 \mapsto f_1(x_1, x_2)$ est constante ($\equiv f_1(x_1, 0)$).

(b) $\forall (x_1, x_2) \in D : \quad f(x_1, x_2) = f(x_1, 0) + \varepsilon\, x_2 \dfrac{\partial f}{\partial x_1}(x_1, 0).$

Preuve :

La proposition (a) découle immédiatement du théorème 1.4.

Pour montrer la proposition (b), il suffit de voir que:

$$\begin{aligned} f_2(x_1, x_2) &= f_2(x_1, 0) + \int_0^{x_2} \frac{\partial f_2}{\partial x_2}(x_1, t)\, dt \\ &= f_2(x_1, 0) + \int_0^{x_2} \frac{\partial f_1}{\partial x_1}(x_1, t)\, dt = f_2(x_1, 0) + x_2 \frac{\partial f_1}{\partial x_1}(x_1, 0) \end{aligned}$$

Ainsi, $f(x_1, x_2) = \underbrace{(f_1(x_1, 0) + \varepsilon\, f_2(x_1, 0))}_{f(x_1, 0)} + \varepsilon\, x_2 \underbrace{\left(\frac{\partial f_1}{\partial x_1}(x_1, 0) + \varepsilon \frac{\partial f_2}{\partial x_1}(x_1, 0)\right)}_{\frac{\partial f}{\partial x_1}(x_1, 0)}$. ∎

Il en découle aussitôt le corollaire suivant:

Corollaire 1.6

Si I est un ouvert de $\boldsymbol{R}$ et $g : I \mapsto \Delta$ est une fonction de classe $\mathcal{C}^2(I)$. Alors, il existe un prolongement Δ-différentiable unique $f : I \times \boldsymbol{R} \mapsto \Delta$ de g qui s'exprime par:

$$f(x,y) = g(x) + \varepsilon\, y\, g'(x)\,; \quad \forall (x,y) \in I \times \boldsymbol{R} \qquad \text{(Eq.1.15)}$$

1.3.5 Prolongement canonique dual d'une fonction de variable réelle

Définition 1.3

Soit f une fonction de variable réelle et à valeur dans $\boldsymbol{R}$ de classe $\mathcal{C}^2$ sur l'intérieur[6] de son domaine de définition $V = \overset{o}{Dom(f)}$. Nous appellerons prolongement canonique dual de la fonction f et nous noterons $\widetilde{f}$, son unique prolongement Δ-différentiable sur V défini par:

$$\widetilde{f}(x) = f(x)\,,\ \forall x \in Dom(f) \qquad \text{(Eq.1.16)}$$

L'existence et l'unicité du prolongement canonique dual découlent directement du corollaire 1.6.

Théorème 1.7

Soient I et J deux ouverts de $\boldsymbol{R}$ et $f : I \mapsto J$ un difféomorphisme de classe $\mathcal{C}^2$ de I sur J. Notons $g = f^{-1}$, le difféomorphisme réciproque. Alors, les prolongements canoniques $\widetilde{f} : I \times \boldsymbol{R} \mapsto J \times \boldsymbol{R}$ et $\widetilde{g} : J \times \boldsymbol{R} \mapsto I \times \boldsymbol{R}$ sont des bijections réciproques l'une de l'autre.

Preuve :

En fait, il suffit de vérifier que:

$$\widetilde{g} \circ \widetilde{f} = \widetilde{f} \circ \widetilde{g} = id$$

En effet, soit $x = x_1 + \varepsilon\, x_2$ un nombre dual appartenant à $J \times \boldsymbol{R}$ ($\subseteq \Delta$), montrons par exemple que: $\widetilde{f} \circ \widetilde{g}(x) = x$

- D'abord, comme $f \circ g(u) = u$, alors $(f \circ g)'(u) = 1$ pour tout réel u. Donc d'après la formule de la dérivée de la composée de deux fonctions numériques, nous pouvons écrire:

$$f'(g(u))\, g'(u) = 1\,; \quad \forall u \in \boldsymbol{R} \qquad \text{(Eq.1.17)}$$

- En suite, puisque $\widetilde{g}(x) = g(x_1) + \varepsilon\, x_2\, g'(x_1)$, alors:

$$\widetilde{f}(\widetilde{g}(x)) = f(g(x_1)) + \varepsilon\, x_2\, g'(x_1)\, f'(g(x_1)) \qquad \text{(Eq.1.18)}$$

- En fin, en portant la relation (Eq.1.17) dans la relation (Eq.1.18), on obtient le résultat recherché.

La démonstration de $\widetilde{g} \circ \widetilde{f} = id$ se fait de la même façon. Ainsi, le théorème est démontré. ∎

[6] Rappelons que l'intérieur d'un ensemble est par définition le plus grand ouvert contenu dans cet ensemble.

Exemples et applications

Dans cet ordre d'idées, nous avons écrit le module `dualstruc.map` [Chapitre 7] pour le système de calcul formel Maple V [Fig. 1.1]. Ce module rend possible le calcul symbolique sur les nombres et quantités duaux. Ainsi, des calculs très complexes peuvent être menés de façon purement symbolique. Les exemples suivants ont été obtenus grâce à ce module:

Le lecteur pourra vérifier alors que les prolongements canoniques:

- $\widetilde{f_1}$ et $\widetilde{f_3}$ sont réciproques l'une de l'autre.
- $\widetilde{f_2}$ et $\widetilde{f_4}$ sont réciproques l'une de l'autre.
- La multiplication de $\widetilde{f_5}$ par elle même (au sens de la multiplication de ses valeurs duales) est équivalente à l'application identique sur $\boldsymbol{R}^+ \times \boldsymbol{R}$ ($\subset \Delta$).
- $\widetilde{f_6}$ est involutive.

Tableau 1.1 : *Exemples de prolongements canoniques duaux*

i	*fonction* f_i	*Prolongement canonique dual* $\widetilde{f_i}$
1	$x \mapsto sin\, x$	$(x_1, x_2) \mapsto sin(x_1) + \varepsilon\, x_2\, cos(x_1)$
2	$x \mapsto cos\, x$	$(x_1, x_2) \mapsto cos(x_1) - \varepsilon\, x_2\, sin(x_1)$
3	$x \mapsto arcsin\, x$	$(x_1, x_2) \mapsto arcsin(x_1) + \frac{\varepsilon\, x_2}{\sqrt{1-{x_1}^2}}$
4	$x \mapsto arccos\, x$	$(x_1, x_2) \mapsto arccos(x_1) - \frac{\varepsilon\, x_2}{\sqrt{1-{x_1}^2}}$
5	$x \mapsto \sqrt{x}$	$(x_1, x_2) \mapsto \sqrt{x_1} + \frac{\varepsilon\, x_2}{2\sqrt{x_1}}$
6	$x \mapsto x^{-1}$	$(x_1, x_2) \mapsto {x_1}^{-1} - \frac{\varepsilon\, x_2}{{x_1}^2}$
7	$h \circ g$	$(x_1, x_2) \mapsto h(g(x_1)) + \varepsilon\, x_2\, g'(x_1)\, h'(g(x_1))$
8	$f_4 \circ f_1$	$(x_1, x_2) \mapsto arccos(sin(x_1)) - \frac{\varepsilon\, x_2\, cos(x_1)}{\sqrt{1-sin(x_1)^2}}$
9	$f_2 \circ f_6 \circ f_3$	$(x_1, x_2) \mapsto cos(arcsin(x_1)^{-1}) + \frac{\varepsilon\, x_2\, sin(arcsin(x_1)^{-1})}{arcsin(x_1)^2 \sqrt{1-{x_1}^2}}$

1.4 Vecteurs duaux

1.4.1 Construction des vecteurs duaux – Définitions

Soit E l'espace vectoriel euclidien réel de dimension 3. Considérons l'ensemble $E \times E$ muni de l'addition:

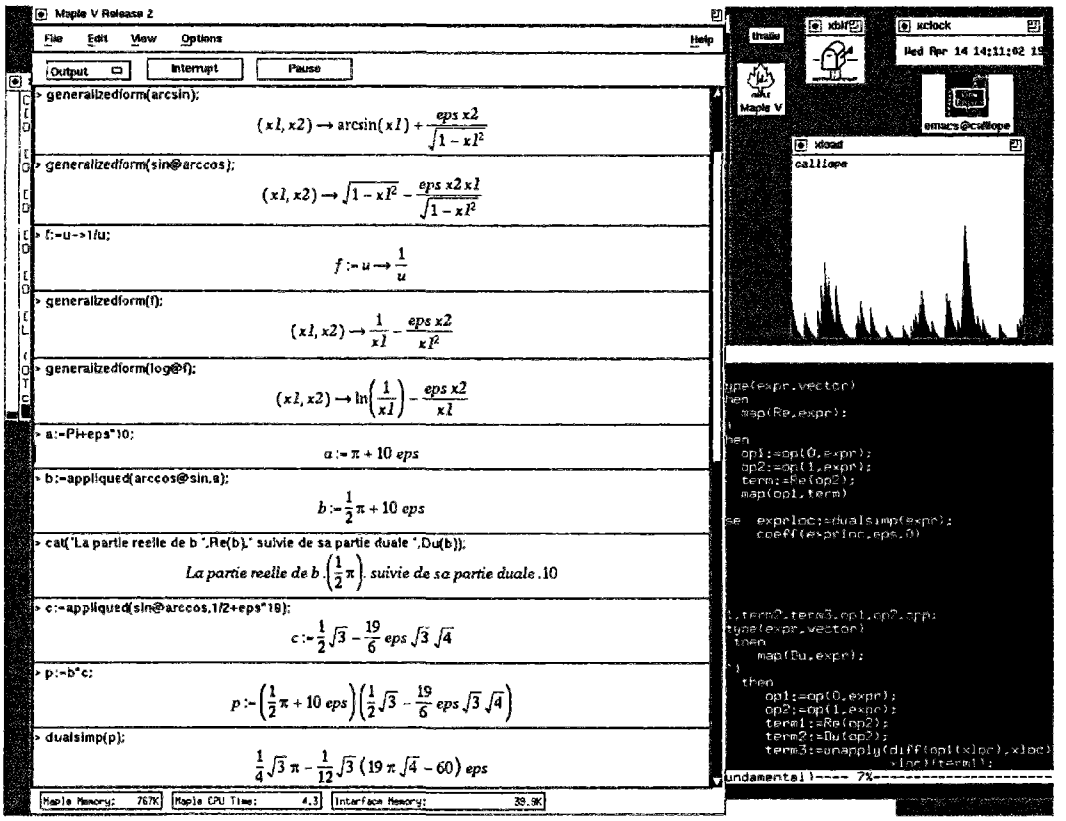
$$(t,u) + (v,w) = \quad (t+v, u+w)\,; \quad \forall t,u,v,w \in E \qquad \text{(Eq.1.19)}$$

et la multiplication par un scalaire dual:

$$(\lambda_1, \lambda_2)\,(u,v) = \quad (\lambda_1\, u, \lambda_1\, v + \lambda_2\, u)\,; \quad \forall \lambda_1, \lambda_2 \in \boldsymbol{R}\,,\ \forall u,v \in E \qquad \text{(Eq.1.20)}$$

L'ensemble $E \times E$ reste stable par ces deux opérations. De plus, il est facile de voir que cet ensemble possède une structure de Δ–module [7]. L'ensemble $E \times E$ muni des lois d'addition

[7] Etant donné un anneau commutatif A, on appelle module sur A ou A–module, un ensemble M muni d'une structure algébrique définie par la donnée:



Actuellement, le module `dualstruc.map` [Chapitre 7] compte un grand nombre de nouvelles fonctions, entièrement implémentées, dans le langage Maple (construction symbolique des nombres duaux, parties réelle et duale d'expressions duales, diverses types de simplifications sur tout genre d'expressions duales, ...etc) pour la manipulation symbolique des objets et structures algébriques duaux (fonctions de variables duale, construction automatique des prolongements canoniques, vecteurs duaux, matrices duales, ...etc). Dans la fenêtre (Maple V Release 2) de la figure ci-dessus, on présente une session de travail sur Maple où l'on expérimente certaines fonctions développées dans le module `dualstruc.map`.

Fig. 1.1: Session de travail sur Maple avec le module `dualstruc.map`

(a) d'une loi de groupe commutatif dans M (notée additivement dans la suite);

(b) d'une loi d'action $(\alpha, x) \mapsto \alpha\, x$, dont le domaine d'action est l'anneau A, et telle que:

(b1) $\alpha\,(x+y) = (\alpha\, x) + (\alpha\, y)\,;\ \forall\, x, y \in M,\ \alpha \in A$;

et de multiplication, définies respectivement par les relations (Eq.1.19) et (Eq.1.20), sera dit dans la suite Δ–module des vecteurs duaux et noté $\widetilde{E}$. De plus, pour des raisons semblables à celles invoquées au paragraphe 1.2.1:

- L'ensemble $\{(u,0)/\, u \in E\}$ est un $\boldsymbol{R}$–espace vectoriel qui peut être identifié à l'espace vectoriel E, à un isomorphisme d'espaces vectoriels près.
- $\widetilde{E}$ peut être identifié à $E \oplus \varepsilon E$ à un isomorphisme de Δ–module près et les éléments de $\widetilde{E}$ pourront être notés symboliquement:

$$(u,v) = 1_\Delta\,(u,0) + \varepsilon\,(v,0) \equiv u + \varepsilon\, v\,; \quad \forall (u,v) \in \widetilde{E} \tag{Eq.1.21}$$

- De la même façon, on peut définir les parties réelle et duale d'un vecteur dual par:

$$Re\,(u,v) = u \text{ et } Du\,(u,v) = v\,; \quad \forall (u,v) \in \widetilde{E} \tag{Eq.1.22}$$

Proposition 1.8
Le Δ–module $\widetilde{E}$ est libre et de dimension trois.

Preuve :

Pour montrer que $\widetilde{E}$ est libre, il suffit de voir que E est un espace vectoriel de dimension trois et que toute base de E est à la fois libre et génératrice du module $\widetilde{E}$ (i.e. $\widetilde{E} \equiv \Delta \otimes_{\boldsymbol{R}} E$. Ainsi, $\widetilde{E}$ possède des bases). De plus, si $w = u + \varepsilon\, v$ est un vecteur dual et si (u_1, u_2, u_3) et (v_1, v_2, v_3) sont les systèmes de coordonnées associés respectivement aux vecteurs u et v relativement à une base donnée de E, alors $(u_1 + \varepsilon\, v_1, u_2 + \varepsilon\, v_2, u_3 + \varepsilon\, v_3)$ est le système de coordonnées représentant w par rapport à cette même base.

Maintenant, toutes les bases de $\widetilde{E}$ sont équipotentes, puisque l'anneau Δ est commutatif et en conséquence, le module $\widetilde{E}$ est libre de dimension 3. ∎

1.4.2 Opérations élémentaires sur les vecteurs duaux

Dans cette section nous allons voire que toutes les opérations définies sur l'espace vectoriel E s'étendent, quand elles sont définies, de manière naturelle au Δ–module $\widetilde{E}$.

1.4.2.a Produit scalaire canonique - Vecteurs duaux orthogonaux

Le Δ–module libre $\widetilde{E}$ peut être muni d'une forme bilinéaire symétrique dite produit scalaire dual canonique, notée par un point et définie par la relation:

$$u.v = Re(u).Re\,(v) + \varepsilon\,(Re\,(u).Du\,(v) + Du\,(u).Re\,(v)) \tag{Eq.1.23}$$

(b2) $(\alpha+\beta)\,x = (\alpha\,x) + (\beta\,x)\,;\ \forall \alpha, \beta \in A,\ x \in M$;

(b3) $\alpha\,(\beta\,x) = (\alpha\,\beta)\,x\,;\ \forall \alpha, \beta \in A,\ x \in M$;

(b4) $1_A\,x = x\,;\ \forall x \in M$.

Ainsi, les ensembles Δ, Δ^3 ou, d'une façon générale, Δ^p sont des exemples types de Δ–module.

où le symbole "." -point- dans $u.v$ représente le produit scalaire dans $\widetilde{E}$ et dans les expressions $Re(u).Re(v)$, $Re(u).Du(v)$ et $Du(u).Re(v)$ le produit scalaire dans E.

Maintenant, si (u_1, u_2, u_3) et (v_1, v_2, v_3) désignent les systèmes de coordonnées duales associés aux vecteurs u et v relativement à une base orthonormée de $\widetilde{E}$ (nous allons caractériser dans la suite de telles bases), alors on peut montrer facilement que:

$$u.v = \sum_{i=1}^{3} u_i v_i \qquad \text{(Eq.1.24)}$$

Deux vecteurs duaux u et v seront dits orthogonaux dans le module $\widetilde{E}$ pour le produit scalaire dual, si $u.v = 0$; c'est à dire aussi:

$$Re(u).Re(v) = 0 \text{ et } Re(u).Du(v) = -Du(u).Re(v) \qquad \text{(Eq.1.25)}$$

1.4.2.b Pseudo-norme duale

Le but que nous nous fixons dans ce paragraphe est de définir une extension canonique de la norme euclidienne (définie sur E) aux vecteurs duaux dont la partie réelle n'est pas nulle. Il n'est pas question d'espérer vérifier les inégalités triangulaire ou de Cauchy-Schwartz pour une telle extension. En effet, cela n'est pas possible puisqu'on ne peut pas définir de relations d'ordre totales sur les scalaires duaux.

Ainsi, pour tout vecteur dual u de $\widetilde{E}$, tel que $Re(u) \neq \vec{0}$, on va poser:

$$\| u \|_{\Delta} = \sqrt{u.u} \qquad \text{(Eq.1.26)}$$

où le symbole "$\sqrt{\ }$" désigne le prolongement canonique dual de la fonction racine carrée. Maintenant, si $u = \vec{0}$, on convient donc de poser:

$$\| u \|_{\Delta} = 0 \qquad \text{(Eq.1.27)}$$

Par un calcul simple, on peut montrer que la relation (Eq.1.26) s'écrit:

$$\| u \|_{\Delta} = \| Re(u) \| + \varepsilon \frac{Re(u).Du(u)}{\| Re(u) \|} \quad \text{si } \| Re(u) \| \neq 0 \qquad \text{(Eq.1.28)}$$

où "$\| \ . \ \|$" est la norme euclidienne habituelle associée à la structure d'espace vectoriel euclidien de E. La relation (Eq.1.28) montre bien que les bases orthonormées de E sont, à un isomorphisme près, des bases orthonormées de $\widetilde{E}$.

1.4.2.c Produit vectoriel dual

Le module libre $\widetilde{E}$ étant de dimension 3, nous allons définir le produit vectoriel dual comme étant une application bilinéaire antisymétrique de $\widetilde{E} \times \widetilde{E} \to \widetilde{E}$:

$$u \times v = Re(u) \times Re(v) + \varepsilon (Re(u) \times Du(v) + Du(u) \times Re(v)); \quad \forall u, v \in \widetilde{E} \qquad \text{(Eq.1.29)}$$

où le symbole "×" dans le membre de droite désigne l'application $\boldsymbol{R}$–bilinéaire du produit vectoriel sur E et "×" dans le membre de gauche est son extension à $\widetilde{E}$. Ainsi, si (u_1, u_2, u_3) et (v_1, v_2, v_3) désignent les coordonnées associées aux vecteurs duaux u et v relativement à une base orthonormée de $\widetilde{E}$, on peut remarquer que:

$$u \times v \equiv \quad [u_2\, v_3 - u_3\, v_2 \quad u_3\, v_1 - u_1\, v_3 \quad u_1\, v_2 - u_2\, v_1]^t \qquad \text{(Eq.1.30)}$$

1.4.2.d Propriétés générales

Les relations (Eq.1.24) et (Eq.1.30) montrent bien que les opérations sur le Δ–module libre $\widetilde{E}$ héritent de toutes les propriétés immédiates des opérations du produit scalaire et du produit vectoriel de l'espace vectoriel euclidien orienté E. Ainsi, si u, v et w sont trois vecteurs duaux arbitraires et α un nombre dual quelconque, les relations suivantes sont vérifiées:

(a) La relation du double produit vectoriel:

$$(u \times v) \times w = \quad (u.w)\, v - (v.w)\, u \qquad \text{(Eq.1.31)}$$

(b) L'identité de Jacobi:

$$u \times (v \times w) + v \times (w \times u) + w \times (u \times v) = \vec{0} \qquad \text{(Eq.1.32)}$$

(c) La relation du produit mixte:

$$u.(v \times w) = \quad w.(u \times v) = \quad v.(w \times u) \qquad \text{(Eq.1.33)}$$

(d) $u \times v$ est orthogonal à la fois à u et à v, pour le produit scalaire dual.

1.5 Matrices duales

1.5.1 Construction

Dans la suite, nous allons nous intéresser, pour des raisons d'ordre pratique, aux matrices duales carrées d'ordre 3×3. Nous pouvons construire la structure de matrices sur les nombres duaux, à partir de $\mathcal{M}_3(\boldsymbol{R}) \times \mathcal{M}_3(\boldsymbol{R})$ muni de certaines lois internes et externes, par une démarche analogue à celle qui nous a permis de construire les nombres et les vecteurs duaux. Toutefois, donnons une description succincte des étapes de construction des opérations usuelles sur de telles matrices et de l'identification de l'ensemble $\mathcal{M}_3(\boldsymbol{R}) \times \mathcal{M}_3(\boldsymbol{R})$ à l'ensemble des matrices à composantes duales $\mathcal{M}_3(\Delta)$:

- Addition de matrices: $(A, B) + (C, D) = (A + C, B + D)\,; \quad \forall A, B, C, D \in \mathcal{M}_3(\boldsymbol{R})$,
- Multiplication de matrices: $(A, B)\,(C, D) = (A\,C, A\,D + B\,C)\,; \quad \forall A, B, C, D \in \mathcal{M}_3(\boldsymbol{R})$,
- Multiplication de matrices par des scalaires duaux:

$$(a, b)\,(A, B) = (a\,A, a\,B + b\,A)\,; \quad \forall A, B \in \mathcal{M}_3(\boldsymbol{R}), \forall a, b \in \boldsymbol{R}$$

- Action des matrices duales à gauche sur les vecteurs duaux:

$$(A,B)\,(u,v) = (A\,u, A\,v + B\,u)\,; \quad \forall A,B \in \mathcal{M}_3(\boldsymbol{R}), \forall u,v \in E$$

- L'ensemble $\mathcal{M}_3(\boldsymbol{R}) \times \mathcal{M}_3(\boldsymbol{R})$ des matrices duales, ainsi construit, possède la structure algébrique de Δ-module libre de dimension 9 (i.e. 9 composantes duales indépendantes). Il sera noté $\mathcal{M}_3(\Delta)$ et dit ensemble des matrices duales d'ordre 3×3.

- L'ensemble $\{(A,0)/\,A \in \mathcal{M}_3(\boldsymbol{R})\} \equiv \{[(a_{ij},0)]_{i,j=1..3}\,/\,a_{ij} \in \boldsymbol{R},\, \forall i,j = 1..3\}$ est un sous-anneau de $\mathcal{M}_3(\Delta)$ isomorphe à $\mathcal{M}_3(\boldsymbol{R})$. Ainsi, dans la suite, nous pourrons identifier cet ensemble à $\mathcal{M}_3(\boldsymbol{R})$.

- $\mathcal{M}_3(\Delta)$ peut être identifié à $\mathcal{M}_3(\boldsymbol{R}) \oplus \varepsilon\,\mathcal{M}_3(\boldsymbol{R})$ par l'isomorphisme de Δ-modules:

$$(A,B) \mapsto 1_\Delta\,A + \varepsilon\,B \equiv A + \varepsilon\,B \equiv [a_{ij} + \varepsilon\,b_{ij}]_{i,j=1..3}$$

Ainsi, nous pouvons étendre la notion de parties réelle et duale aux objets mathématiques de type matrice duale. Les opérateurs Re et Du peuvent, donc, être définis de $\mathcal{M}_3(\Delta)$ dans $\mathcal{M}_3(\boldsymbol{R})$ à partir des parties réelle et duale de nombres duaux:

$$A = [A_{ij}]_{i,j=1..3} \quad \Rightarrow \quad \begin{cases} Re\,(A) = & [Re\,(a_{ij})]_{i,j=1..3} \\ Du\,(A) = & [Du\,(a_{ij})]_{i,j=1..3} \end{cases} \qquad \text{(Eq.1.34)}$$

Ainsi, toute matrice duale peut être décomposée, de façon unique, en fonction de ses parties réelle et duale:

$$A = Re\,(A) + \varepsilon Du\,(A); \quad \forall A \in \mathcal{M}_3(\Delta) \qquad \text{(Eq.1.35)}$$

En conséquence, les opérations usuelles sur les matrices duales peuvent être réécrites dans une notation symbolique qui peut être retenue comme représentation canonique pour leur implémentation algorithmique dans des processus de calcul formel:

(a) La somme de deux matrices duales se réduit à la somme de matrices réelles par:

$$A + B = Re(A) + Re\,(B) + \varepsilon\,(Du\,(A) + Du\,(B)) \qquad \text{(Eq.1.36)}$$

(b) Le produit de composition matricielle, de deux matrice duales, se réduit au produit de matrices réelles par la relation:

$$A\,B = Re\,(A)\,Re\,(B) + \varepsilon\,(Re\,(A)\,Du\,(B) + Du\,(A)\,Re\,(B)) \qquad \text{(Eq.1.37)}$$

(c) Le produit d'une matrice par un scalaire dual se traduit par:

$$\lambda\,A = Re(\lambda)\;Re\,(A) + \varepsilon\;(Re\,(\lambda)\;Du\,(A) + Du\,(\lambda)\;Re\,(A)) \qquad \text{(Eq.1.38)}$$

(d) L'action d'une matrice duale sur un vecteur dual se résume à:

$$A\,v = \quad Re\,(A)\,Re\,(v) + \varepsilon\,(Re\,(A)\,Du\,(v) + Du\,(A)\,Re\,(v)) \qquad \text{(Eq.1.39)}$$

(e) La transposée d'une matrice duale se traduit par la somme des transposées de ses parties réelle et duale:

$$A^t = \quad Re\,(A)^t + \varepsilon\,Du\,(A)^t \qquad \text{(Eq.1.40)}$$

1.5.2 Inversion de matrices carrées duales

Le but de ce paragraphe est de caractériser les matrices carrées duales inversibles et d'exprimer l'inverse, de telles matrices, quand il existe.

Théorème 1.9

Pour qu'une matrice A soit inversible dans le module $\mathcal{M}_3(\Delta)$, il faut et il suffit que la matrice $Re\,(A)$ soit inversible (c'est à dire $\det Re\,(A) \neq 0$). Dans ces conditions la matrice A^{-1} s'exprime par la relation:

$$A^{-1} = \quad Re\,(A)^{-1} - \varepsilon\,Re\,(A)^{-1}\,Du\,(A)\,Re\,(A)^{-1} \qquad \text{(Eq.1.41)}$$

Preuve :

Pour que $A \in \mathcal{M}_3(\Delta)$ soit inversible, il faut et il suffit qu'il existe $B \in \mathcal{M}_3(\Delta)$, telle que $A\,B = B\,A = \mathbb{1}$ (où $\mathbb{1}$ est la matrice unité); c'est à dire, d'une part:

$$Re\,(B)\,Re\,(A) = \quad Re\,(A)\,Re\,(B) = \quad \mathbb{1} \qquad \text{(Eq.1.42)}$$

et, d'autre part:

$$\begin{cases} Re\,(A)\,Du\,(B) + Du\,(A)\,Re\,(B) = \quad 0 \\ Re\,(B)\,Du\,(A) + Du\,(B)\,Re\,(A) = \quad 0 \end{cases} \qquad \text{(Eq.1.43)}$$

La relation (Eq.1.42) signifie que si A est inversible, $Re(A)$ est inversible. Réciproquement si $Re(A)$ est inversible et si les relations (Eq.1.42) et (Eq.1.43) sont satisfaites par B alors:

$$B = \quad Re\,(A)^{-1} - \varepsilon\,Re\,(A)^{-1}\,Du\,(A)\,Re\,(A)^{-1}$$

ce qui montre le théorème. ∎

1.5.3 Matrice duale orthogonale

Dans ce paragraphe, nous souhaitons caractériser les matrices duales orthogonales.

Définition 1.4

Une matrice carrée duale sera dite orthogonale, si et seulement si elle est inversible et vérifie la condition d'orthogonalité:

$$A^t\,A = \quad \mathbb{1} \qquad \text{(Eq.1.44)}$$

Cela veut dire que $A^{-1} = A^t$. Donc, d'après le théorème 1.9, d'une part, la matrice $Re\,(A)$ est une matrice orthogonale au sens classique $Re\,(A)^{-1} = Re\,(A)^t$ et, d'autre part:

$$Re\,(A)\,Du\,(A)^t + Du\,(A)\,Re\,(A)^t = 0 \qquad \text{(Eq.1.45)}$$

Théorème 1.10

Soit A une matrice carrée duale. Si A est orthogonale, alors, il existe deux vecteurs réels u et v, tels que:

$$A = (\mathbb{1} + \varepsilon\,\check{u})\,Re\,(A) = Re(A)\,(\mathbb{1} + \varepsilon\,\check{v}) \qquad \text{(Eq.1.46)}$$

où $\check{u}$ est la matrice antisymétrique définie par: $\check{u}\,w = u \times w$ pour tout $w \in E$.

Preuve :

La relation (Eq.1.45) exprime que la matrice $Du(A)\,Re\,(A)^t$ est une matrice réelle antisymétrique. Par la suite, il existe un vecteurs $u \in E$ tel que: $\check{u} = Du\,(A)\,Re\,(A)^t$ et posant $v = Re\,(A)^t\,u\,Re\,(A)$, on obtient le résultat recherché. ∎

1.6 Conclusion

Dans le domaine de la modélisation, on est souvent tenté par des généralisations et des changements de représentations algébriques successives[8] du problème initial afin de se ramener à des domaines plus riches en outils de calcul symbolique et qui conduisent à une implémentation optimale des algorithmes. Changer la représentation algébrique d'un problème peut conduire à des solutions de problèmes plus généraux que le problème initial sans compliquer pour autant la représentation syntaxique du modèle utilisé. Ainsi, dans les problèmes qui nous intéressent pour la description de modèles assez généraux, la structure de module sur l'anneau des nombres duaux semble être particulièrement pratique pour le calcul symbolique. Cependant, contrairement à ce que l'on pourrait croire, même si les axiomes d'un module sont identiques à ceux d'un espace vectoriel, au changement près du corps de base en un anneau, et même si les principales définitions d'algèbre linéaire sont valables dans les modules, les propriétés algébriques d'un module peuvent différer substantiellement de celles d'un espace vectoriel. En particulier, des propriétés intuitives de la géométrie ou de l'existence d'une norme peuvent être en défaut dans un module. En effet, faute d'une justification mathématique rigoureuse des résultats, on ne peut affirmer avec certitude la validité des modèles généralisés. Mais, une fois le contexte mathématique de ces extensions est bien défini, il devient naturel de chercher les correspondances qui existent entre les modèles exprimés dans les différentes représentations. Ces correspondances permettraient de relier les structures algébriques aux modèles étudiés. Cette étape fait l'objet du chapitre suivant.

[8] Et souvent excessives si ces généralisations ne sont pas efficacement contrôlées.

Chapitre 2

CONCEPTS ET OUTILS PRELIMINAIRES DE MODELISATION

2.1 Introduction

Plusieurs méthodes existent pour la modélisation mathématique des problèmes de la mécanique du solide rigide. La prise en compte de l'évolution des outils mathématiques dans la modélisation de tels problèmes est un paramètre important dans la conception et l'élaboration de modèles généraux. Le calcul numérique et symbolique de ces modèles fait généralement appel à la notion de changement de repères pour décrire les déplacements relatifs des différents éléments du système mécanique les uns par rapport aux autres. La construction de repères conduit à la définition de systèmes de coordonnées pour la paramétrisation du groupe des déplacements et de l'espace des configurations.

Ce chapitre est consacré à la présentation de l'outil mathématique qui permet une telle description dans la théorie des groupes et algèbres de Lie. Nous allons par la suite associer une structure de module à l'algèbre de Lie des torseurs. Pour définir les systèmes de coordonnées, nécessaires à la manipulation numérique et symbolique des êtres mathématiques intervenant dans la modélisation, nous allons introduire la notion de familles fondamentales de l'algèbre de Lie des torseurs $\mathfrak{D}$ comme une alternative à la notion classique de repères (i.e. systèmes d'axes) de l'espace géométrique affine $\mathcal{E}$ et nous montrerons que ces deux notions sont équivalentes. Notre choix est justifié pour deux raisons essentielles:

- L'automatisation du calcul symbolique à un degré d'abstraction assez élevé. Cela va permettre de faire des manipulations symboliques sur les familles fondamentales; donc directement sur les repères associés.
- L'ergonomie de la représentation syntaxique des modèles, pour permettre une meilleure transparence du calcul symbolique et une plus simple localisation des simplifications du calcul à un degré d'abstraction assez élevé.

2.1.1 Notion du groupe des déplacements euclidiens

En se plaçant dans le cadre de la géométrie euclidienne, la théorie mathématique des groupes peut être appliquée à l'ensemble des déplacements de corps rigides (i.e. déplacements euclidiens). Cet ensemble forme un groupe de Lie [J. Favard [FAVA-57], J-M. Hervé [HERV-78], A. Karger & J. Novak [KARG-85], D. Chevallier [CHEV-86] & [CHEV-91]] que nous noterons dans la suite $\mathbb{D}$. Dire que l'ensemble $\mathbb{D}$ est un groupe de Lie signifie que cet ensemble est muni d'une structure de groupe, d'une structure de variété différentielle et que la composition et l'inversion des déplacements sont des opérations différentiables [Annexe A].

Dire qu'un corps rigide $\mathcal{C}$ est contraint à évoluer dans un sous-groupe de $\mathbb{D}$, signifie que ce corps est soumis à des liaisons d'origine mécanique. Nous reviendrons sur cette idée, pour développer plus amplement la notion de contraintes lors de l'étude des liaisons mécaniques [Chapitre 3].

2.1.2 Notion de configuration d'un corps rigide

La notion de configuration du corps rigide $\mathcal{C}$ est un élément de modélisation assez abstrait. Généralement, une configuration $\boldsymbol{s}$ est donnée par rapport à une référence $\boldsymbol{r}$, éventuellement virtuelle. On caractérise, souvent, cette notion par la donnée de la position et l'orientation du corps. L'ensemble des configurations d'un corps solide forme une variété différentielle $\mathcal{S}$. L'opération du groupe des déplacements $\mathbb{D}$ à gauche sur l'ensemble $\mathcal{S}$ peut être définie de façon complètement abstraite par les trois axiomes suivants [D. Chevallier [CHEV-86]]:

(a_1) $\forall \boldsymbol{A}, \boldsymbol{B} \in \mathbb{D}$, $\forall \boldsymbol{s} \in \mathcal{S}$: $\boldsymbol{A} \bullet (\boldsymbol{B} \bullet \boldsymbol{s}) = (\boldsymbol{A}\,\boldsymbol{B}) \bullet \boldsymbol{s}$.

(a_2) $\forall \boldsymbol{s} \in \mathcal{S}$: $\boldsymbol{e} \bullet \boldsymbol{s} = \boldsymbol{s}$.

(a_3) $\forall \boldsymbol{r}$ (fixé) $\in \mathcal{S}$: $\boldsymbol{A} \mapsto \boldsymbol{A} \bullet \boldsymbol{r}$ est une bijection de $\mathbb{D}$ sur $\mathcal{S}$.

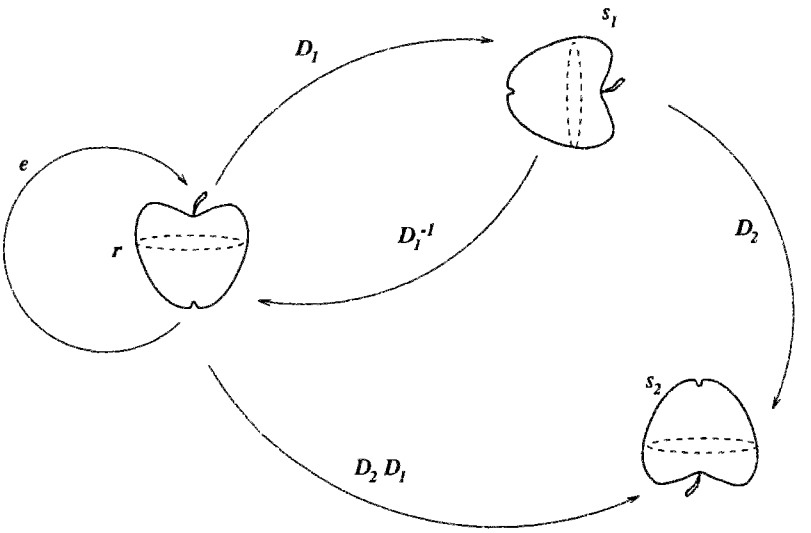
Ces trois axiomes traduisent le fait que l'espace $(\mathcal{S}, \mathbb{D}, \bullet)$ est un espace homogène sans points fixe[1]. Nous allons montrer dans ce chapitre qu'il est possible d'identifier la variété $\mathcal{S}$, à un isomorphisme près, à l'ensemble $\mathfrak{F}$ des familles fondamentales donnant ainsi une interprétation de $\mathcal{S}$ par des éléments de $\mathfrak{d}$ et une interprétation de l'action "$\bullet$" par l'action naturelle du groupe de Lie $\mathbb{D}$ sur l'algèbre de Lie des torseurs $\mathfrak{d}$.

2.2 Algèbre de Lie des torseurs

2.2.1 Rappels – Définitions – Notations

Notons $\mathcal{E}$ l'espace géométrique affine et E l'espace vectoriel associé. Dans cette section, nous allons, d'une part rappeler, la terminologie de base et les propriétés immédiates de la notion de torseur et, d'autre part, définir les notations dans lesquels nous allons travailler:

[1] Le lecteur pourra se rapporter à l'annexe A.



L'action de "déplacement" est l'opération qui permet un changement de configurations d'un corps tout en restant dans l'hypothèse du solide rigide. Ainsi:

- Un changement de configurations $[r \rightarrow s_2]$ est indépendant du chemin suivi pour atteindre la configuration finale s_2.
- Si on applique à un corps rigide $\mathcal{C}$ un déplacement D_1, pour remettre $\mathcal{C}$ dans sa configuration initiale; il suffit de lui appliquer l'opération inverse ${D_1}^{-1}$;
- La composée $D_2\,D_1$ de deux déplacements D_2 et D_1 est également un déplacement;
- On convient qu'il existe un déplacement e qui laisse fixe toute configuration de solide rigide. C'est l'élément neutre du groupe des déplacements.

Ainsi, le groupe de Lie des déplacements euclidiens, pourra être interprété, comme le groupe de transformations de la variété différentielle $\mathcal{S}$ des configurations spatiales d'un corps rigide.

Fig. 2.1: Action du groupe $\mathbb{D}$ sur l'ensemble des configurations $\mathcal{S}$

- On appelle champ de vecteurs toute application X de $\mathcal{E}$ sur E.
- Un champ de vecteurs X est équiprojectif, s'il vérifie la condition:

$$X(M).\overrightarrow{MN} = X(N).\overrightarrow{MN}; \quad \forall M,\, N \in \mathcal{E}$$

- Pour tout champ équiprojectif (ou torseur) X, il existe un vecteur ω_X tel que:

$$X(M) = X(N) + \omega_X \times \overrightarrow{NM}; \quad \forall M,\, N \in \mathcal{E}$$

Les vecteurs ω_X et $X(M)$ sont dits les éléments de réduction du torseur X au point M. Il sont dits respectivement vecteur résultant et moment du torseur X. De plus, la donnée de la paire $\{\omega_X, X(M)\}$ en un point M de $\mathcal{E}$ caractérise de façon unique le torseur X. Nous allons noter, dans la suite, $\mathfrak{D}$ l'ensemble de tous les torseurs.

- La magnitude du torseur X est la norme de son vecteur résultant (i.e $\mathrm{m}(X) = \| \omega_X \|$) pour la norme "$\| \, . \, \|$" associée au produit scalaire "." sur E.

- On appelle forme bilinéaire de Klein des torseurs X et Y et on note $[X/Y]$, la forme $\boldsymbol{R}$-bilinéaire symétrique définie par: $[X/Y] = \omega_X.Y(M) + X(M).\omega_Y$.

- Les torseurs élémentaires sont les torseurs isotropes pour la forme de Klein (i.e. tels que $[X/X] = 0$). Il existe deux types de torseurs isotropes pour la forme de Klein:

  - On appelle torseur constant ou couple un torseur dont le vecteur résultat est nul. On notera $\mathfrak{t}$ l'ensemble des torseurs constant.

  - On appelle glisseur un torseur tel qu'il existe un point $M \in \mathcal{E}$ qui annule son moment. On notera $\mathfrak{g}$ l'ensemble de tous les glisseurs de $\mathfrak{D}$.

- En tout point M de l'espace affine $\mathcal{E}$, tout torseur X se décompose de façon unique en un torseur constant et un glisseur. De plus, soit $\mathfrak{c}_M = \{X \in \mathfrak{D} / X(M) = 0\}$, l'ensemble $\mathfrak{c}_M$ est un sous-espace vectoriel de $\mathfrak{D}$ sur $\boldsymbol{R}$ et $\mathfrak{D} = \mathfrak{c}_M \oplus \mathfrak{t}$. Cette décomposition est dite décomposition canonique de $\mathfrak{D}$ au point M.

- Si $X \in \mathfrak{D} \setminus \mathfrak{t}$, on appelle pas[2] du torseur X l'application de $\mathfrak{D}$ dans $\boldsymbol{R}$ définie par:

$$\mathrm{p}(X) = \frac{[X/X]}{2\,\mathrm{m}(X)^2}$$

- Si $X \in \mathfrak{D} \setminus \mathfrak{t}$, on appelle axe central du torseur X que nous noterons Λ_X l'ensemble des points de $\mathcal{E}$ tels que les éléments de réductions de X soient colinéaires.

- On note opérateur Ω, l'opérateur défini dans $\mathfrak{D}$, qui à un torseur X associe le torseur constant de moment: $\Omega X\,(M) = \omega_X\,; \quad \forall M \in \mathcal{E}$.

- L'image et le noyau de l'opérateur Ω sont confondus: $Im\,\Omega = Ker\,\Omega = \mathfrak{t}$ (i.e $\Omega^2 = 0$).

2.2.2 Structure d'algèbre de Lie de l'espace vectoriel des torseurs

On associe, généralement, au groupe de Lie des déplacements euclidiens $\mathbb{D}$, l'algèbre de Lie des torseurs $\mathfrak{D}$ munie du crochet de Lie.

[2]Traduction anglaise: "pitch".

Théorème et définition 2.1

L'application bilinéaire $(X,Y) \mapsto [X,Y]$ *définie par*[3]*:*

$$[X,Y](M) = \omega_X \times Y(M) + X(M) \times \omega_Y \; ; \; \forall M \in \mathcal{E} \qquad \text{(Eq.2.1)}$$

muni l'espace vectoriel $\mathfrak{D}$ *de la structure d'algèbre de Lie.*

Preuve :

$\mathfrak{D}$ est un espace vectoriel. Il est évident que le crochet "[,]", défini par la relation (Eq.2.1), est une application bilinéaire, antisymétrique. Un calcul simple permet de montrer que le crochet de Lie "[,]" vérifie l'identité de Jacobi (i.e. $[X,[Y,Z]] + [Y,[Z,X]] + [Z,[X,Y]] = 0; \quad \forall X, Y, Z \in \mathfrak{D}$). ∎

2.2.3 Multiplication des torseurs par les nombres duaux

Considérons maintenant l'espace vectoriel $\mathcal{L}(\mathfrak{D})$ des opérateurs linéaires sur l'algèbre de Lie $\mathfrak{D}$. La notion d'opérateurs cylindriques[4] va établir une interprétation, en terme d'opérateurs linéaires, de la multiplication de torseurs par des nombres duaux.

Définition 2.1

Nous appellerons opérateur cylindrique tout opérateur linéaire de $\mathcal{L}(\mathfrak{D})$ *de la forme:*

$$\Sigma_{\alpha,a} = \alpha\, id + a\,\Omega \qquad \text{(Eq.2.2)}$$

où α *et* a *sont des réels quelconques.*

Soit $\Sigma(\mathfrak{D})$ l'ensemble des opérateurs cylindriques. Il est clair que cet ensemble est un sous anneau et un $\boldsymbol{R}$–sous-espace vectoriel (i.e. sous-algèbre) de $\mathcal{L}(\mathfrak{D})$. De plus $\Sigma(\mathfrak{D})$ et Δ sont isomorphes (par l'isomorphisme d'anneaux $\Sigma_{\alpha,a} \mapsto \alpha + \varepsilon\, a$).

Ainsi, la multiplication d'un torseur X par un nombre dual $\alpha + \varepsilon\, a$, s'identifie à l'image du torseur X par l'opérateur cylindrique $\Sigma_{\alpha,a}$. On écrira:

$$(\alpha + \varepsilon\, a)\, X \equiv \Sigma_{\alpha,a}\, X = \alpha\, X + a\,\Omega\, X$$

2.3 Représentation duale des torseurs

2.3.1 Opérateur de réduction d'un torseur en un point

Dans ce paragraphe, nous allons définir l'opérateur de réduction d'un torseur en un point et les propriétés qu'implique une telle opération sur la manipulation des torseurs.

[3]Dans la suite, nous préférerons la relation:

$$[X,Y](M) = \omega_X \times Y(M) - \omega_Y \times X(M)$$

à la relation (Eq.2.1), pour des raisons de symétrie du calcul symbolique sur les structures duales. En effet, cette remarque fera l'objet de la proposition 2.2 [page 36].

[4]J'ai choisi l'appellation *opérateur cylindrique* parce que, comme on va le voir plus tard, les générateurs infinitésimaux des déplacements cylindriques (composition d'une rotation et d'une translation de même axe) se ramènent à l'image d'un glisseur unitaire par de tels opérateurs.

Définition 2.2

Soit O un point quelconque de l'espace euclidien affine $\mathcal{E}$. Nous appellerons opérateur de réduction au point O, l'application définie par:

$$\begin{array}{rccl} \tau_O : & \mathfrak{D} & \rightarrow & \widetilde{E} \\ & X & \mapsto & \omega_X + \varepsilon\, X(O) \end{array}$$

Pour alléger les notations nous désignerons par X_O, le vecteur dual image du torseur X par l'application τ_O.

Proposition 2.2

Soit O un point quelconque de l'espace euclidien affine $\mathcal{E}$. Alors:

i) L'opérateur de réduction τ_O est un isomorphisme de Δ–modules.

ii) Si "$\times$" désigne le produit vectoriel sur le Δ–module $\widetilde{E}$. Alors:

$$\tau_O([X,Y]) = \tau_O(X) \times \tau_O(Y)\ ;\ \forall X, Y \in \mathfrak{D} \qquad \text{(Eq.2.3)}$$

Preuve :

Montrons d'bord la proposition *i*). Il est évident que $\mathfrak{D}$ est un Δ–module et que τ_O est un homomorphisme de Δ–modules. Montrons que cet homomorphisme est bijectif:

Soit un vecteur dual $u \in \widetilde{E}$. Nous savons que tout torseur est défini d'une manière unique par la donnée de ses éléments de réduction en un point de l'espace euclidien affine $\mathcal{E}$. Ainsi, il existe un unique torseur X de $\mathfrak{D}$ tel que: $\omega_X = Re(u)$, $X(O) = Du(u)$. Cela signifie que l'homomorphisme τ_O est bijectif. Ce qui montre la relation la proposition *i*).

Ensuite, pour montrer la proposition *ii*), il suffit de voir que d'après la relation (Eq.2.1):

$$\begin{cases} Re\,[X,Y]_O = \omega_{[X,Y]} = \omega_X \times \omega_Y = Re\,X_O \times Re\,Y_O \\ Du\,[X,Y]_O = [X,Y](O) = \omega_X \times Y(O) + X(O) \times \omega_Y = Re\,X_O \times Du\,Y_O + Du\,X_O \times Re\,Y_O \end{cases}$$

Ce qui montre la relation la proposition *ii*. ∎

Ainsi, d'après les propositions 1.8 [page 24] et 2.2.*i*, nous avons le résultat suivant:

Corollaire 2.3

Le Δ–module $(\mathfrak{D}, +, .)$ est libre de dimension trois.

2.3.2 Produit scalaire dual sur le Δ–module des torseurs

Théorème et définition 2.4

La structure de Δ–module libre de $\mathfrak{D}$ peut être munie d'une forme Δ–bilinéaire:

$$\{X/Y\} = \omega_X.\omega_Y + \varepsilon\ [X/Y] \qquad \text{(Eq.2.4)}$$

qui sera dite produit scalaire dual sur $\mathfrak{D}$.

Preuve :

Il est clair que "$\{./.\}$" est une forme R–bilinéaire. De plus elle est symétrique, puisque le produit scalaire sur E des vecteurs résultants et la forme de Klein le sont (somme de deux

fonctions symétriques). La linéarité par rapport à l'addition étant évidente, il reste à montrer que si $x = \alpha + \varepsilon\, a$, alors, $\{x\, X/Y\} = x\, \{X/Y\}$. En effet:

$$\{x\, X/Y\} = \alpha\, \{X/Y\} + a\, \{\varepsilon\, X/Y\} = \alpha\, \{X/Y\} + a\, \varepsilon\, \omega_X\, \omega_Y = x\, \{X/Y\}$$

d'où le résultat. ∎

Proposition 2.5

Soient X, Y, Z trois torseurs de $\mathfrak{D}$, alors, le double crochet de Lie s'exprime par la relation:

$$[[X,Y],Z] = \{X/Z\}\, Y - \{Y/Z\}\, X \qquad \text{(Eq.2.5)}$$

La démonstration de cette proposition découle directement de la relation (Eq.1.31) en prenant pour vecteurs duaux les image par τ_O des torseurs X, Y, Z en un point quelconque $O \in \mathcal{E}$ [page 26].

2.3.3 Pseudo-norme sur le Δ–module des torseurs

Nous pouvons, ainsi définir une certaine pseudo-norme duale (pseudo-norme euclidienne), issue du produit scalaire dual sur l'ensemble $\mathfrak{D} \setminus \mathfrak{t}$ (i.e. l'ensemble des torseurs non constants) par:

$$\| X \|_{\mathfrak{d}} = \begin{cases} \sqrt{\{X/X\}}\,; & \forall X \in \mathfrak{D} \setminus \mathfrak{t} \\ 0 \quad \text{si } X = 0 \end{cases} \qquad \text{(Eq.2.6)}$$

Proposition 2.6

Si X est un torseur non constant, alors, la pseudo-norme euclidienne de X s'exprime par la relation[5]*:*

$$\| X \|_{\mathfrak{d}} = \mathrm{m}(X)\, [1 + \varepsilon\, \mathrm{p}(X)]\,;\ \forall X \in \mathfrak{D} \setminus \mathfrak{t} \qquad \text{(Eq.2.7)}$$

La démonstration est simple, elle consiste à appliquer la relation (Eq.1.15) avec la fonction "$\sqrt{\ }$" et le nombre dual $x = \{X/X\}$.

2.3.4 Produit mixte dual de torseurs

Théorème et définition 2.7

Nous appellerons produit mixte dual la forme Δ–tri-linéaire alternée définie par:

$$(X,Y,Z) \mapsto \{X/[Y,Z]\}$$

Cette application vérifie:

$$\{X/[Y,Z]\} = \{Z/[X,Y]\} = \{Y/[Z,X]\} \qquad \text{(Eq.2.8)}$$

[5] Cette pseudo-norme peut être obtenue aussi à partir de la pseudo-norme euclidienne sur Δ^3 [page 1.4.2] par la relation:

$$\| X \|_{\mathfrak{d}} = \| X_O \|_{\Delta}\,;\ \forall O \in \mathcal{E}$$

Preuve :

La démonstration de la partie théorème de cet énoncé est triviale. Elle découle des propriétés du produit mixte sur $\widetilde{E}$, compte tenu de la proposition 2.2.*i* puisque $\{X/[Y,Z]\} = X_O \times (Y_O \times Z_O)$, pour tout point $O \in \mathcal{E}$. ∎

2.3.5 Torseurs unitaires et torseurs orthogonaux

Définition 2.3

Nous appellerons torseur unitaire tout torseur $X \in \mathfrak{D}$, tel que:

$$\| X \|_{\mathfrak{d}} = 1 \qquad \text{(Eq.2.9)}$$

Il paraît donc clair qu'un torseur unitaire est nécessairement un glisseur -de pas égal à zéro- de magnitude égale à un.

A ce stade du développement, on peut logiquement se poser les questions suivantes:

- Quelles sont les relations entre les bases ortho-normales du Δ–module libre $\mathfrak{D}$ et les bases de la structure de R–espace vectoriel de $\mathfrak{D}$.
- Comment sont construites les bases ortho-normales "directes" du Δ–module libre $\mathfrak{D}$ de dimension trois (muni de "$\{./.\}$" et du produit mixte)?
- Comment est construit le "groupe orthogonal" de $\mathfrak{D}$, muni du produit intérieur dual "$\{./.\}$"?

C'est le propos des deux sections suivantes.

2.4 Famille fondamentale de l'algèbre de Lie des torseurs

Nous allons définir dans ce paragraphe une notion qui peut être interprétée de plusieurs façons[6] selon la structure associée à $\mathfrak{D}$. Cette notion va permettre, ainsi, la paramétrisation de l'algèbre de Lie des torseurs $\mathfrak{D}$ et, par la suite, la paramétrisation du groupe des déplacements $\mathbb{D}$.

Théorème 2.8

Considérons l'algèbre de Lie des torseurs $\mathfrak{D}$:

(a) Si $X \in \mathfrak{D} \setminus \mathfrak{t}$ et $\mathfrak{p}(X) = 0$, alors, X s'annule sur son axe central Λ_X.

(b) Si $X, Y \in \mathfrak{D} \setminus \mathfrak{t}$ tels que $\mathfrak{p}(X) = \mathfrak{p}(Y) = 0$, alors:

$$[X/Y] = 0 \iff (\Lambda_X \text{ et } \Lambda_Y \text{ sont coplanaires.}) \qquad \text{(Eq.2.10)}$$

en particulier si $\omega_X \times \omega_Y \neq 0$ (i.e. $[X,Y] \notin \mathfrak{t}$), les axes Λ_X et Λ_Y se coupent.

[6] Voir les théorème 2.9 et 2.10 [page 39–40].

Preuve :

La proposition *(a)* est évidente. Il reste à montrer la proposition *(b)*:

or, dire que les pas des torseurs X et Y sont nuls, signifie que tous les deux sont des glisseurs. Considérons un point O fixe de l'axe Λ_Y, d'après la proposition *(a)*:

$$\forall M \in \Lambda_X : \quad [X/Y] = \omega_X.(\omega_Y \times \overrightarrow{OM}) = 0$$

i.e. $\overrightarrow{OM}.(\omega_X \times \omega_Y) = 0$. Donc, nous avons deux cas à examiner:

- si $\omega_X \times \omega_Y = 0$, alors, les axes Λ_X et Λ_Y sont parallèles, ils sont donc coplanaires.
- si $\omega_X \times \omega_Y \neq 0$, alors, les axes Λ_X et Λ_Y ne peuvent pas être parallèles.
  De plus $\overrightarrow{OM}.(\omega_X \times \omega_Y) = 0$ signifie que les vecteurs $\overrightarrow{OM}$, ω_X et ω_Y sont coplanaires. i.e. le point M appartient au plan (O, ω_Y, ω_X) (qui contient l'axe Λ_Y). Ceci étant pour tout point M de l'axe Λ_X, les axes Λ_X et Λ_Y sont donc coplanaires.

Ce qui montre le théorème. ∎

Définition 2.4

Nous appellerons famille fondamentale de l'algèbre de Lie $\mathfrak{D}$; un triplet de torseurs (ξ, η, ζ) non nuls, tels qu'ils vérifient les trois conditions suivantes:

$$[\xi, \eta] = \zeta\,,\ [\eta, \zeta] = \xi\,,\ [\zeta, \xi] = \eta \qquad \text{(Eq.2.11)}$$

Dans la suite les conditions (Eq.2.11) seront dites conditions d'ortho-orientation de la famille fondamentale.

Définition 2.5

On considère le Δ-module libre $\mathfrak{D}$ muni du produit "$\{./.\}$" et du produit mixte. Soit $\mathfrak{f} = (\xi, \eta, \zeta)$ une famille ortho-normale de $\mathfrak{D}$, $\mathfrak{f}$ sera dite directe, si et seulement si:

$$Re\ \{[\xi, \eta]/\zeta\} > 0 \qquad \text{(Eq.2.12)}$$

Théorème 2.9

Soit un triplet de torseurs (ξ, η, ζ) de $\mathfrak{D}$. Les propriétés suivantes sont équivalentes:

(a) (ξ, η, ζ) est une base du Δ-module libre $\mathfrak{D}$;

(b) $(\xi, \eta, \zeta, \Omega\xi, \Omega\eta, \Omega\zeta)$ est une base du $\boldsymbol{R}$-espace vectoriel $\mathfrak{D}$;

(c) $(\Omega\xi, \Omega\eta, \Omega\zeta)$ est une partie libre sur $\boldsymbol{R}$ (i.e. $(\Omega\xi, \Omega\eta, \Omega\zeta)$ est une base de $\mathfrak{t}$)[7].

Preuve :

[7] Notons que cela veut dire aussi que, tout simplement, $(\omega_\xi, \omega_\eta, \omega_\zeta)$ est une base de l'espace vectoriel euclidien E.

- $\boxed{(a) \Longrightarrow (b)}$
  Soient $\alpha, \beta, \gamma, a, b$ et c six nombres réels, la relation: $\alpha\,\xi + \beta\,\eta + \gamma\,\zeta + a\,\Omega\,\xi + b\,\Omega\,\eta + c\,\Omega\,\zeta = 0$ équivaut à: $(\alpha + \varepsilon\,a)\,\xi + (\beta + \varepsilon\,b)\,\eta + (\gamma + \varepsilon\,c)\,\zeta = 0$ (une combinaison linéaire à cœfficients dans Δ). Si la proposition (a) est vérifiée, cela implique: $\alpha + \varepsilon\,a = \beta + \varepsilon\,b = \gamma + \varepsilon\,c = 0$ et donc: $\alpha = \beta = \gamma = a = b = c = 0$.

- $\boxed{(b) \Longrightarrow (c)}$
  Cette implication est triviale, puisque toute partie d'une base est une famille libre.

- $\boxed{(c) \Longrightarrow (a)}$
  La combinaison linéaire: $(\alpha + \varepsilon\,a)\,\xi + (\beta + \varepsilon\,b)\,\eta + (\gamma + \varepsilon\,c)\,\zeta = 0$ implique (en multipliant les deux membres par ε): $\varepsilon\,\alpha\,\xi + \varepsilon\,\beta\,\eta + \varepsilon\,\gamma\,\zeta = \alpha\,\Omega\,\xi + \beta\,\Omega\,\eta + \gamma\,\Omega\,\zeta = 0$ et, d'après (c) $\alpha = \beta = \gamma = 0$. Il reste alors: $a\,\Omega\,\xi + b\,\Omega\,\eta + c\,\Omega\,\zeta = 0$, d'où par le même raisonnement: $a = b = c = 0$.

Ce qui prouve l'équivalence entre les proposition (a), (b) et (c). ∎

Théorème 2.10

Soit un triplet de torseurs (ξ, η, ζ) de $\mathfrak{D}$. Les propriétés suivantes sont équivalentes:

(a) (ξ, η, ζ) est une famille fondamentale de $\mathfrak{D}$;

(b) (ξ, η, ζ) est une base directe du Δ-module libre $\mathfrak{D}$;

(c) il existe un repère ortho-normé direct $(O; \vec{i}, \vec{j}, \vec{k})$ tel que les torseurs ξ, η et ζ soient déterminés par:

$$\left\{ \begin{array}{ll} \xi(M) = & \vec{i} \times \overrightarrow{OM}, \\ \eta(M) = & \vec{j} \times \overrightarrow{OM}, \\ \zeta(M) = & \vec{k} \times \overrightarrow{OM} \end{array} \right. ; \forall M \in \mathcal{E} \qquad \text{(Eq.2.13)}$$

De plus, dans ces conditions:

$$\{[\xi, \eta]\,/\zeta\} = 1 \qquad \text{(Eq.2.14)}$$

Preuve :

- $\boxed{(a) \Longrightarrow (b)}$
  A priori, (ξ, η, ζ) étant une base du Δ-module libre $\mathfrak{D}$, il existe a, b, $c \in \Delta$ tels que: $[\xi, \eta] = a\,\xi + b\,\eta + c\,\zeta$ cette base étant ortho-normale, alors:

  $$\{\xi/[\xi, \eta]\} = a\{\xi/\xi\} = 0\,,\ \{\eta/[\xi, \eta]\} = b\{\eta/\eta\} = 0\,,\ \{\zeta/[\xi, \eta]\} = c\{\zeta/\zeta\} = c\,.$$

  ainsi, $[\xi, \eta] = c\,\zeta$ avec $c = \{\zeta/[\xi, \eta]\}$
  et plus généralement $[\xi, \eta] = c\,\zeta$; $[\eta, \zeta] = c\,\xi$; $[\zeta, \xi] = c\,\eta$ (avec le même c).

On en déduit alors: $[\eta,[\xi,\eta]] = c\,[\eta,\zeta] = c^2\,\xi$ et d'après la formule (Eq.2.5) du double crochet de Lie:

$$[\eta,[\xi,\eta]] = \{\eta/\eta\}\,\xi - \{\eta/\xi\}\,\eta = \xi$$

finalement: $\xi = c^2\,\xi$ d'où $c^2 = 1$ et comme $Re\,c = Re\,\{\zeta/[\xi,\eta]\} > 0$, alors: $c = 1$.

- $\boxed{(b) \Longrightarrow (a)}$
  La relation (b) entraîne, d'une part, que: $\{\xi/\eta\} = \{\eta/\zeta\} = \{\zeta/\xi\} = 0$
  et, d'autre part: $\{\xi,\xi\} = \{\eta,\eta\} = \{\zeta,\zeta\} = \{\zeta/\{\xi,\eta\}\} = c$
  et

$$\begin{aligned}\{\xi,\xi\} &= \{[\xi,\eta]/[\xi,\eta]\}\\ &= \{[\xi,[\eta,\xi]]/\eta\}\\ &= \{\xi,\xi\}\,\{\eta,\eta\} - \{\xi/\eta\}\,\{\xi/\eta\}\\ &= \{\xi,\xi\}\,\{\eta,\eta\}\end{aligned}$$

  d'où $c = c^2$ et l'on a deux solutions possibles (dans Δ); à savoir $c = 0$ et $c = 1$.
  Or, si $c = 0$ alors: $\xi = \eta = \zeta = 0_{\mathfrak{d}}$ (ce qui est exclu par hypothèse). Donc, $c = 1$ et en conséquence la famille (ξ,η,ζ) forme une base ortho-normale du Δ–module libre $\mathfrak{D}$.

- $\boxed{(a) \Longrightarrow (c)}$
  Les relations exprimant l'ortho-normalité de la famille (ξ,η,ζ) impliquent que:
  - les vecteurs sommes: $\vec{i} = \omega_\xi\,; \vec{j} = \omega_\eta\,; \vec{k} = \omega_\zeta$ forment une base ortho-normale de l'espace vectoriel réel de dimension trois E;
  - les parties duales: $[\xi/\xi] = [\eta/\eta] = [\zeta/\zeta] = 0$. Cela implique que les axes Λ_ξ, Λ_η et Λ_ζ sont deux à deux sécants et deux à deux orthogonaux. Ces axes se coupent en un même point $O \in \mathcal{E}$ et $\xi(O) = \eta(O) = \vec{0}$, or nous avons montrer que $(\vec{i}\times\vec{j}).\vec{k} = Re\,\{\zeta/[\xi,\eta]\} > 0$, donc, le repère $(O;\vec{i},\vec{j},\vec{k})$ est un repère ortho-normé direct de l'espace euclidien orienté $\mathcal{E}$.
  - $\boxed{(c) \Longrightarrow (a)}$
    La démonstration de cette implication découle d'un calcul immédiat.

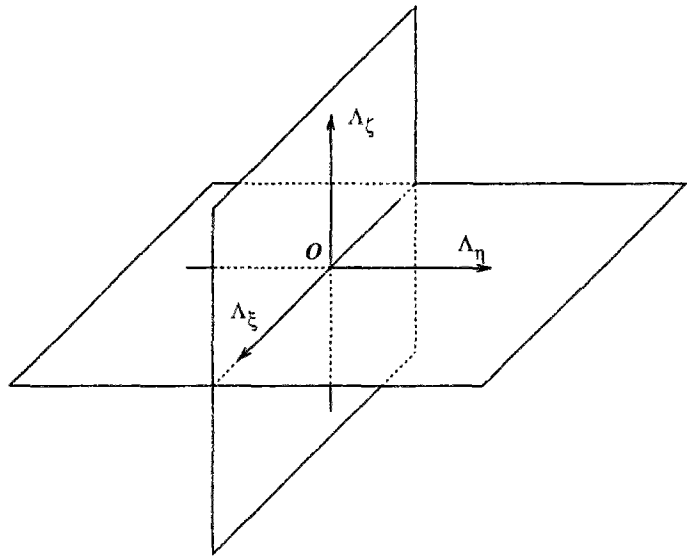
Ce qui montre le théorème. ∎

Définition 2.6

Soit $\mathfrak{f} = (\xi,\eta,\zeta)$ *une famille fondamentale de l'algèbre de Lie* $\mathfrak{D}$. *Nous appellerons origine de* $\mathfrak{f}$ *et nous noterons* $\mathfrak{o}(\mathfrak{f})$ *l'intersection*[8] *des axes* Λ_ξ, Λ_η *et* Λ_ζ.

[8] L'unique point O de $\mathcal{E}$ (Voir le théorème 2.10 [page 40]) tel que:

$$\xi(O) = \eta(O) = \zeta(O) = \vec{0}$$



Cette figure schématise les axes associés à une famille fondamentale. Ces axes forment un trièdre dont l'origine est celui de la famille fondamentale.

Fig. 2.2: Famille fondamentale

2.5 Groupe orthogonal de $\mathfrak{d}$ – Déplacements euclidiens

Le but de cette section est de déterminer le "groupe orthogonal" de $\mathfrak{d}$ et caractériser les déplacements euclidiens en tant qu'opérateurs sur $\mathfrak{d}$.

Théorème 2.11

Soit une application $F : \mathfrak{d} \to \mathfrak{d}$ ensembliste à priori. Les deux propositions suivantes sont équivalentes:

(a) F est un endomorphisme qui conserve la pseudo-norme "$\| \cdot \|_\delta$".

(b) F conserve le produit scalaire dual sur le Δ-module libre $\mathfrak{d}$.

Une telle application est nécessairement bijective et elle détermine une isométrie du Δ-module libre $\mathfrak{d}$.

Preuve :

- $\boxed{(a) \Longrightarrow (b)}$

  Considérons deux torseurs arbitraires X et Y:$\{F(X)+F(Y)/F(X)+F(Y)\} = \| F(X) \|_{\mathfrak{d}}^2 + \| F(Y) \|_{\mathfrak{d}}^2 + 2\{F(X)/F(Y)\}$ et comme F est linéaire, alors:

$$\{F(X)/F(Y)\} = \frac{1}{2}\left[\| F(X)+F(Y) \|_{\mathfrak{d}}^2 - \| F(X) \|_{\mathfrak{d}}^2 - \| F(Y) \|_{\mathfrak{d}}^2\right]$$

$$= \frac{1}{2}\left[\| F(X+Y) \|_{\mathfrak{d}}^2 - \| F(X) \|_{\mathfrak{d}}^2 - \| F(Y) \|_{\mathfrak{d}}^2\right]$$

  et d'après (a) on a: $\{F(X)/F(Y)\} = \frac{1}{2}\left[\| X+Y \|_{\mathfrak{d}}^2 - \| X \|_{\mathfrak{d}}^2 - \| Y \|_{\mathfrak{d}}^2\right] = \{X/Y\}$.

- $\boxed{(b) \Longrightarrow (a)}$

  Soit (ξ, η, ζ) une base ortho-normale du Δ–module libre $\mathfrak{D}$. Comme F conserve le produit scalaire, alors:

$$\begin{cases} \{F(\xi)/F(\xi)\} = \{F(\eta)/F(\eta)\} = \{F(\zeta)/F(\zeta)\} = 1 \\ \{F(\xi)/F(\eta)\} = \{F(\eta)/F(\zeta)\} = \{F(\xi)/F(\zeta)\} = 0 \end{cases} \qquad \text{(Eq.2.15)}$$

  $(F(\xi), F(\eta), F(\zeta))$ est une famille de trois glisseurs unitaires deux à deux orthogonaux, donc elle définit également une base ortho-normale de $\mathfrak{D}$ (cela découle du corollaire 2.3 et du fait que l'anneau Δ est commutatif). Maintenant:

$$\begin{cases} \{F(x\xi+y\eta+z\zeta)/F(\xi)\} = \{x\xi+y\eta+z\zeta/\xi\} = x \\ \{F(x\xi+y\eta+z\zeta)/F(\eta)\} = \{x\xi+y\eta+z\zeta/\eta\} = y \\ \{F(x\xi+y\eta+z\zeta)/F(\zeta)\} = \{x\xi+y\eta+z\zeta/\zeta\} = z \end{cases}$$

  donc $F(x\xi+y\eta+z\zeta) = xF(\xi)+yF(\eta)+zF(\zeta)$ et F est alors Δ–linéaire. Par ailleurs, d'après (b) on a: $\{F(X)/F(X)\} = \{X/X\}$, donc $\| F(X) \|_{\mathfrak{d}} = \| X \|_{\mathfrak{d}}$.

ce qui montre le théorème. ∎

Ce théorème caractérise le groupe orthogonal du Δ–module libre $\mathfrak{D}$.

Corollaire 2.12

Soit $F : \mathfrak{D} \to \mathfrak{D}$ *ensembliste à priori. Les deux propositions suivantes sont équivalentes:*

(a) F *conserve le produit scalaire dual et le produit mixte sur le* Δ*-module libre* $\mathfrak{D}$.

(b) Il existe un déplacement euclidien $\boldsymbol{D}$ *de* $\mathbb{D}$, *tel que:*

$$F = \boldsymbol{D}_* \qquad \text{(Eq.2.16)}$$

Preuve :

- $\boxed{(b) \Longrightarrow (a)}$

  Elle découle du théorème précédent.

- $\boxed{(a) \Longrightarrow (b)}$

  F conserve le produit mixte, en d'autres termes:

$$\text{si } Re\{[X,Y]/Z\} > 0 \text{ alors } Re\{[F(X),F(Y)]/F(Z)\} = Re\{[X,Y]/Z\} > 0$$

  et compte tenu du théorème 2.11, cela implique que F transforme les repères ortho-normés directs de $\mathcal{E}$ en d'autres repères ortho-normés directs de $\mathcal{E}$. Donc, il existe un déplacement $\boldsymbol{D}$ tel que $F = \boldsymbol{D}_*$

∎

2.6 Action du groupe des déplacements à gauche sur $\mathfrak{F}$

Notons par $\mathfrak{F}$ l'ensemble de toutes les familles fondamentales. Nous allons établir dans ce paragraphe des propriétés qui présentent un intérêt capital pour tous les aspects de modélisation et les interprétations ultérieures.

Proposition 2.13

Le groupe $\mathbb{D}$ opère $\mathfrak{F}$ à gauche[9] *par l'action "$\diamond$" définie par:*

$$\boldsymbol{D} \diamond \mathfrak{f} = (\boldsymbol{D}_*\xi, \boldsymbol{D}_*\eta, \boldsymbol{D}_*\zeta) : \forall \boldsymbol{D} \in \mathbb{D} \text{ et } \forall \mathfrak{f} = (\xi,\eta,\zeta) \in \mathfrak{F} \qquad \text{(Eq.2.17)}$$

Preuve :

Il suffit de montrer que d'une part l'ensemble $\mathfrak{F}$ de toutes les familles fondamentales reste stable par l'opération "$\diamond$" et que d'autre part $\boldsymbol{D}$ opère à gauche de $\mathfrak{F}$. En effet:

- Considérons une famille fondamentale $\mathfrak{f} \in \mathfrak{F}$ et un déplacement euclidien $\boldsymbol{D} \in \mathbb{D}$ à priori arbitraires:

$$\begin{cases} [\xi,\eta] = \zeta \\ [\eta,\zeta] = \xi \\ [\zeta,\xi] = \eta \end{cases} \Rightarrow \begin{cases} \boldsymbol{D}_*[\xi,\eta] = \boldsymbol{D}_*\zeta \\ \boldsymbol{D}_*[\eta,\zeta] = \boldsymbol{D}_*\xi \\ \boldsymbol{D}_*[\zeta,\xi] = \boldsymbol{D}_*\eta \end{cases} \Rightarrow \begin{cases} [\boldsymbol{D}_*\xi, \boldsymbol{D}_*\eta] = \boldsymbol{D}_*\zeta \\ [\boldsymbol{D}_*\eta, \boldsymbol{D}_*\zeta] = \boldsymbol{D}_*\xi \\ [\boldsymbol{D}_*\zeta, \boldsymbol{D}_*\xi] = \boldsymbol{D}_*\eta \end{cases}$$

  ce qui montre que $\boldsymbol{D} \diamond \mathfrak{f}$ est bien une famille fondamentale.

- Par ailleurs:

  - soit $\mathfrak{f} \in \mathfrak{F}$ (quelconque), si e désigne l'élément neutre du groupe $\mathbb{D}$, alors, $e_* = \mathbb{1}$ et, donc, $e \diamond \mathfrak{f} = \mathfrak{f}$.

  - pour tout déplacement $\boldsymbol{D}$, l'application $\mathfrak{f} \mapsto \boldsymbol{D} \diamond \mathfrak{f}$ est bijective. En effet, son inverse est l'application $\mathfrak{f} \mapsto \boldsymbol{D}^{-1} \diamond \mathfrak{f}$.

[9] Voir Annexe A [pp. 214-215].

- Posons, maintenant, $\mathfrak{f} = (\xi, \eta, \zeta)$ et considérons deux déplacements $\boldsymbol{A}$ et $\boldsymbol{B}$ quelconques du groupe $\mathbb{D}$:

$$\begin{aligned} \boldsymbol{A} \diamond (\boldsymbol{B} \diamond \mathfrak{f}) &= \boldsymbol{A} \diamond (\boldsymbol{B}_* \xi, \boldsymbol{B}_* \eta, \boldsymbol{B}_* \zeta) \\ &= (\boldsymbol{A}_*(\boldsymbol{B}_* \xi), \boldsymbol{A}_*(\boldsymbol{B}_* \eta), \boldsymbol{A}_*(\boldsymbol{B}_* \zeta)) \\ &= ((\boldsymbol{A}_* \circ \boldsymbol{B}_*) \xi, (\boldsymbol{A}_* \circ \boldsymbol{B}_*) \eta, (\boldsymbol{A}_* \circ \boldsymbol{B}_*) \zeta) \\ &= ((\boldsymbol{A}\,\boldsymbol{B})_* \xi, (\boldsymbol{A}\,\boldsymbol{B})_* \eta, (\boldsymbol{A}\,\boldsymbol{B})_* \zeta) \\ &= (\boldsymbol{A}\,\boldsymbol{B}) \diamond \mathfrak{f} \end{aligned}$$

ce qui montre que $\mathbb{D}$ opère à gauche de $\mathfrak{F}$ par l'opération "$\diamond$". ∎

Théorème 2.14
L'ensemble $(\mathfrak{F}, \mathbb{D}, \diamond)$ *est un espace homogène sans points fixes.*

∎ **Preuve :**

Il s'agit de montrer que: $\forall\, \mathfrak{f}_1, \mathfrak{f}_2 \in \mathfrak{F}$, $\exists !\ \boldsymbol{D} \in \mathbb{D}$ tel que: $\mathfrak{f}_2 = \boldsymbol{D} \diamond \mathfrak{f}_1$.

- **Existence:** Ecrivons d'abord: $\mathfrak{f}_1 = (\xi_1, \eta_1, \zeta_1)$ et $\mathfrak{f}_2 = (\xi_2, \eta_2, \zeta_2)$ et considérons l'application $F \in \mathcal{L}(\mathfrak{V})$ définie par sa matrice duale:

$$[F] = \begin{pmatrix} \{\xi_2/\xi_1\} & \{\eta_2/\xi_1\} & \{\zeta_2/\xi_1\} \\ \{\xi_2/\eta_1\} & \{\eta_2/\eta_1\} & \{\zeta_2/\eta_1\} \\ \{\xi_2/\zeta_1\} & \{\eta_2/\zeta_1\} & \{\zeta_2/\zeta_1\} \end{pmatrix}$$

L'application F est visiblement orthogonale pour le produit scalaire "$\{./.\}$" (sa matrice est duale orthogonale), donc F conserve le produit scalaire dual et le produit mixte dual. Donc, compte tenu du corollaire 2.12, il existe un déplacement $\boldsymbol{D} \in \mathbb{D}$ tel que: $F = \boldsymbol{D}_*$. De plus, $F(\xi_1) = \{\xi_2/\xi_1\}\,\xi_1 + \{\xi_2/\eta_1\}\,\eta_1 + \{\xi_2/\zeta_1\}\,\zeta_1 = \xi_2$. De même, $F(\eta_1) = \eta_2$ et $F(\zeta_1) = \zeta_2$. Donc, $\mathfrak{f}_2 = \boldsymbol{D} \diamond \mathfrak{f}_1$

- **Unicité:** Supposons qu'il existe deux déplacements $\boldsymbol{D}_1$ et $\boldsymbol{D}_2$ tels que:

$$\mathfrak{f}_2 = \boldsymbol{D}_1 \diamond \mathfrak{f}_1 = \boldsymbol{D}_2 \diamond \mathfrak{f}_1$$

Donc: $(\boldsymbol{D}_2{}^{-1}\boldsymbol{D}_1) \diamond \mathfrak{f}_1 = \mathfrak{f}_1$, ainsi, $\forall X \in \mathfrak{V}$: $(\boldsymbol{D}_2{}^{-1}\boldsymbol{D}_1)_* X = X$ i.e. $\boldsymbol{D}_2{}^{-1}\boldsymbol{D}_1 = e$.

Ce qui achève la démonstration. ∎

2.7 Isomorphie de $(\mathbb{S}, \mathbb{D}, \bullet)$ et $(\mathfrak{F}, \mathbb{D}, \diamond)$

L'idée que nous développons dans cette section consiste à adjoindre à une configuration s donnée du solide, une famille fondamentale $\mathfrak{f}$ qui en soit rigidement solidaire. Cela est justifié par le fait que $(S, \mathbb{D}; \bullet)$ et $(\mathfrak{F}, \mathbb{D}; \diamond)$ sont des espaces homogènes sans points fixes opérés par le même groupe $\mathbb{D}$ et donc isomorphes [Annexe A, paragraphe A.4.3, page 215]. En effet,

i) on peut définir une application invariante (ou équivariante) $s \mapsto \mathfrak{f}_s$ de $\mathcal{S}$ dans $\mathfrak{F}$ telle que:

$$\mathfrak{f}_{D \bullet s} = \boldsymbol{D} \diamond \mathfrak{f}_s \qquad \text{(Eq.2.18)}$$

ii) comme les deux espaces sont homogènes sans points fixes, une telle application est bijective (c'est un isomorphisme d'espaces opérés par le groupe $\mathbb{D}$).

iii) si une configuration de référence $\boldsymbol{r} \in \mathcal{S}$ et une famille $\mathfrak{f} \in \mathfrak{F}$, telles que $\mathfrak{f} = \mathfrak{f}_r$, sont données, alors, il existe un unique isomorphisme, de $(\mathcal{S}, \mathbb{D}; \bullet)$ sur $(\mathfrak{F}, \mathbb{D}; \diamond)$, caractérisé par la relation (Eq.2.18).

Ainsi, l'action de "lier une famille fondamentale à un solide rigide" consiste tout simplement à donner un isomorphisme de $(\mathcal{S}, \mathbb{D}; \bullet)$ sur $(\mathfrak{F}, \mathbb{D}; \diamond)$. Cette façon de voir les choses peut être généralisée à tous les types d'objets "liés à un solide rigide".

2.8 Notion de mouvement de solide rigide – Familles mobiles

Pour décrire en cinématique le mouvement d'un solide ou d'un mécanisme en général, nous allons définir la notion de familles mobiles qui peut être vue comme une abstraction de la notion de repères mobiles[10]. Les calculs associés à la notion de familles mobiles peuvent être décrits par l'action du groupe $\mathbb{D}$ sur l'ensemble $\mathfrak{F}$ des familles fondamentales et donc par l'action des matrices duales de $\mathcal{M}_3(\Delta)$ sur le Δ–module libre Δ^3 de dimension 3 (puisque toute famille fondamentale définie un système de coordonnées sur $\widetilde{E}$).

Définition 2.7

Nous appelons famille mobile toute application de $\boldsymbol{R}$ (ou encore de $[t_0, +\infty[$ en prenant t_0 comme origine des temps) dans l'ensemble $\mathfrak{F}$.

En conséquence, si $\mathfrak{f}_0 \in \mathfrak{F}$ est fixé, il existe une application unique $t \mapsto \boldsymbol{D}(t)$ de $\boldsymbol{R}$ dans $\mathbb{D}$, telle que:

$$\mathfrak{f}(t) = \boldsymbol{D}(t) \diamond \mathfrak{f}_0 \qquad \text{(Eq.2.19)}$$

Cela revient au même de décrire un mouvement du solide rigide à partir de sa trajectoire $t \mapsto s(t) \in \mathcal{S}$ telle que pour une configuration de référence fixe r, $s(t)$ est déterminée par l'application $t \mapsto \boldsymbol{D}(t)$ telle que $s(t) = \boldsymbol{D}(t) \bullet \boldsymbol{r}$ (les deux descriptions sont liées par les isomorphismes évoqués plus haut).

2.9 Exponentielle de torseurs – Générateur infinitésimal d'un déplacement

Au point ou nous en sommes de nos développements, de point de vue théorique, il est tout à fait raisonnable de se contenter d'une vision relativement simple de ce qui peut être appelée application

[10]Nous avons démontré qu'une famille fondamentale est équivalente au trièdre ortho-normé direct formé par les axes centraux des glisseurs qui la composent [Chapitre 2, Théorème 2.10, page 40]. Munir une telle famille d'un mouvement donné revient à munir le repère associé du même mouvement.

exponentielle de torseurs. Cette fonction est définie pour tout groupe de Lie et généralise ainsi la notion d'exponentielle habituelle (sur $\mathbb{C}$) ou encore la fonction exponentielle d'opérateurs. En effet, la théorie générale des groupes et algèbres de Lie permet de définir une telle application à partir de la notion de sous-groupes à un paramètre [Annexe A, paragraphe A.5.2, page 219-220]. Ainsi, si e désigne l'élément neutre du groupe $\mathbb{D}$, pour tout torseur $X \in \mathfrak{D}$; la solution maximale $t \mapsto \Phi_X(t)$ des équations différentielles avec condition initiale:

$$\left[\begin{array}{l} \dfrac{d}{dt}\Phi(t) = X \circ \Phi(t), \quad \Phi(0) = e \\ \\ \text{ou encore} \\ \\ \dfrac{d}{dt}\Phi(t) = \Phi(t) \circ X, \quad \Phi(0) = e \end{array}\right. \qquad \text{(Eq.2.20)}$$

est un sous-groupe à un paramètre engendré par le torseur X. La fonction exponentielle du groupe $\mathbb{D}$ n'est autre que l'application notée "exp" telle que: $exp X = \Phi_X(1)$. Il reste, cependant, que cette définition demeure abstraite et il faut bien passer par une représentation adéquate pour décrire les calculs. En effet, un calcul ne peut se faire qu'en termes d'une représentation opératoire. Pour contourner cette difficulté, nous allons définir l'exponentielle de torseurs à partir de l'exponentielle d'opérateurs de $\mathcal{L}(\mathfrak{D})$. Pour une telle définition, nous nous devons de signaler que nous admettons un certain nombre de résultats qui se démontrent de façon générale dans la théorie des groupes et algèbres de Lie[11] à savoir:

i) Dans le groupe $\mathbb{D}$: $\boldsymbol{A}_* = \boldsymbol{B}_* \iff \boldsymbol{A} = \boldsymbol{B}$.

ii) On admet que:

- $\boldsymbol{D} \mapsto \boldsymbol{D}_*$ coïncide avec la représentation adjointe,
- $(exp X)_* = exp(ad X) = \displaystyle\sum_{n=0}^{\infty} \frac{(ad X)^n}{n!}$.

iii) Soient $\mathfrak{N} = \{X \in \mathfrak{D} / \parallel \omega_X \parallel \notin 2\pi\mathbb{N}^*\}$ et $\mathfrak{N}_0 = \{X \in \mathfrak{D} / \parallel \omega_X \parallel < \pi\}$. La restriction de l'application exponentielle à $\mathfrak{N}$ définit un homéomorphisme local de $\mathfrak{N}$ sur un ouvert de $\mathbb{D}$. Sa restriction à $\mathfrak{N}_0$ est un homéomorphisme de $\mathfrak{N}_0$ sur un voisinage ouvert $\mathcal{U}$ de e dans $\mathbb{D}$. L'image par l'application réciproque de tels homéomorphismes définit la notion de "générateur infinitésimal" d'un déplacement. Dans la théorie générale des groupes et algèbres de Lie on parle aussi, d'application logarithme (définie du groupe de Lie dans l'algèbre de Lie). On appelle alors générateur infinitésimal d'un déplacement son image par l'application logarithme.

[11] Pour la démonstration de ces résultat dans le cadre général de la théorie des groupes et algèbres de Lie le lecteur pourra se reporter, par exemple, à la référence [DIE—71].

2.10 Calcul de la représentation adjointe d'un déplacement

Nous proposons ci-après une méthode pour la détermination analytique de la représentation adjointe d'un déplacement quelconque. Cette méthode est purement intrinsèque et indépendante de tout choix du système de coordonnées.

Lemme 2.15

Soit X un torseur de $\mathfrak{d}$. Alors:

(a) Si $X \in \mathfrak{t}$ (i.e. la magnitude de X est nulle), alors, pour $n \geq 2$:

$$(ad\,X)^n = 0 \qquad \text{(Eq.2.21)}$$

(b) Si $X \in \mathfrak{d} \setminus \mathfrak{t}$ (i.e. la magnitude de X est non nulle), alors:

- *pour $n \geq 1$:*

$$(ad\,X)^{2n} = (-1)^{n-1} \parallel X \parallel_{\mathfrak{d}}^{2n-2} (ad\,X)^2 \qquad \text{(Eq.2.22)}$$

- *pour $n \geq 0$:*

$$(ad\,X)^{2n+1} = (-1)^n \parallel X \parallel_{\mathfrak{d}}^{2n} ad\,X \qquad \text{(Eq.2.23)}$$

Preuve :

- La démonstration est évidente dans le cas où $X \in \mathfrak{t}$.
- Dans le cas où $X \in \mathfrak{d} \setminus \mathfrak{t}$, la démonstration s'établit donc par récurrence, en calculant d'abord $(ad\,X)^3$ et $(ad\,X)^4$. En effet, d'après la relation du double crochet de Lie:

$$(ad\,X)^3\,Y = ad\,X\,([X,[X,Y]]) = ad\,X\,(\{X/Y\}\,X - \{X/X\}\,Y) = -\{X/X\}\,[X,Y]$$

i.e. $(ad\,X)^3 = -\{X/X\}\,ad\,X$.

Ensuite, en multipliant chacun des deux membres de cette égalité par $ad\,X$, on obtient la relation $(ad\,X)^4 = -\{X/X\}\,(ad\,X)^2$. Pour le reste il suffit d'exprimer l'hypothèse de récurrence.

Ce qui achève la démonstration du lemme. ∎

Considérons maintenant un déplacement $\boldsymbol{D}$ (par exemple: le déplacement de passage entre deux configurations de l'organe terminal), à priori, à un ou plusieurs degrés de libertés.

Théorème 2.16

Soit un déplacement $\boldsymbol{D} \in \mathbb{D}$ et notons X son générateur infinitésimal. Alors:

- *Si $X \in \mathfrak{t}$ (i.e. la magnitude de X est nulle), alors:*

$$\boldsymbol{D}_* = \mathbb{1} + ad\,X \qquad \text{(Eq.2.24)}$$

- *Si $X \in \mathfrak{d} \setminus \mathfrak{t}$, alors:*

$$\boldsymbol{D}_* = \mathbb{1} + \frac{sin \parallel X \parallel_{\mathfrak{d}}}{\parallel X \parallel_{\mathfrak{d}}} ad\,X + \frac{(1 - cos \parallel X \parallel_{\mathfrak{d}})}{\parallel X \parallel_{\mathfrak{d}}^2} (ad\,X)^2 \qquad \text{(Eq.2.25)}$$

Preuve :

La représentation adjointe du déplacement $\boldsymbol{D}$ s'écrit: $\boldsymbol{D}_* = \mathbb{1} + \sum_{n=1}^{\infty} \frac{(adX)^n}{n!}$.

- Si $X \in \mathfrak{t}$, la relation (Eq.2.24) est évidente.
- Si $X \in \mathfrak{D} \setminus \mathfrak{t}$, en remplaçant les termes de puissances paires par leur expression dans (Eq.2.22) et les termes de puissances impaires par leur expression dans (Eq.2.23), compte tenu du lemme 2.15 et en regroupant d'une part les termes de facteur commun adX et d'autre part les termes de facteur commun $(adX)^2$, on obtient:

$$\boldsymbol{D}_* = \mathbb{1} + \frac{1}{\| X \|_{\mathfrak{d}}} \left(\sum_{n=0}^{\infty} \frac{(-1)^n \| X \|_{\mathfrak{d}}^{2n+1}}{(2n+1)!} \right) adX + \frac{1}{\| X \|_{\mathfrak{d}}^2} \left(\sum_{n=1}^{\infty} \frac{(-1)^{n-1} \| X \|_{\mathfrak{d}}^{2n}}{(2n)!} \right) (adX)^2$$

Compte tenu des relations (Eq.1.9) et (Eq.1.10) [Chapitre 1, corollaire 1.3, page 18], on obtient le résultat recherché.

Ce qui montre le théorème. ∎

2.11 Déplacements élémentaires – Décomposition canonique des déplacements

Vu leur rôle essentiel dans la modélisation, il nous semble logique de chercher, à se stade du développement, à caractériser les déplacements élémentaires (i.e. translations et rotations). En effet, tout déplacement peut se décomposer en une rotation et une translation suivant un même axe.

Proposition 2.17

$\boldsymbol{T}$ est une translation de vecteur $\vec{u}$ si et seulement, s'il existe un torseur U tel que:

$$\boldsymbol{T} = exp(U) \text{ avec } \mathfrak{t} \ni U \text{ de vecteur } \vec{u} \qquad \text{(Eq.2.26)}$$

Preuve :

Soit $\boldsymbol{T}$ est une translation de vecteur $\vec{u}$, alors: $\forall M \in \mathcal{E} : \overrightarrow{M\,\boldsymbol{T}(M)} = \overrightarrow{\boldsymbol{T}^{-1}(M)\,M} = \vec{u}$,

or, $\forall Y \in \mathfrak{D}, \forall M \in \mathcal{E} : \quad (\boldsymbol{T}_* Y)(M) = Y(\boldsymbol{T}^{-1}(M))$

et comme Y est un champ équiprojectif, il vient:

$$(\boldsymbol{T}_* Y)(M) = Y(M) + \overrightarrow{\boldsymbol{T}^{-1}(M)\,M} \times \omega_Y = Y(M) + \vec{u} \times \omega_Y$$

ceci étant en tout point M de l'espace affine $\mathcal{E}$. Prenons le torseur $U \in \mathfrak{t}$ de vecteur $\vec{u}$. La relation précédente implique: $\boldsymbol{T}_* Y = Y + [U, Y] = (\mathbb{1} + adU)\,Y$. Or, comme le résultat est vérifié pour tout torseur Y de $\mathfrak{D}$, on en déduit (Eq.2.26).

La réciproque est évidente, elle découle directement de la définition d'une translation. Ainsi, elle ne soulève aucune difficulté particulière. ∎

Corollaire 2.18

Soit $\mathfrak{f}$ une famille fondamentale donnée et $\mathcal{R}$ le repère ortho-normé direct qui en est associé[12] *:*

$$\boldsymbol{T} \text{ est une translation de vecteur } \vec{u} \iff \begin{cases} \vec{u}/_{\mathcal{R}} = [u_1\ u_2\ u_3]^t \\ [\boldsymbol{T}_*]/_{\mathfrak{f}} = \begin{pmatrix} 1 & -\varepsilon u_3 & \varepsilon u_2 \\ \varepsilon u_3 & 1 & -\varepsilon u_1 \\ -\varepsilon u_2 & \varepsilon u_1 & 1 \end{pmatrix} \end{cases}$$

Proposition 2.19

Si $X \in \mathfrak{g}$, alors, le déplacement défini par $\boldsymbol{R} = \exp X$ est une rotation d'axe Λ_X.

Preuve :

Soit un glisseur X, alors, il existe un point $A \in \mathcal{E}$ tel que $X(A) = \vec{0}$. De plus l'application $t \mapsto exp(t\,X)(A)$ est la solution de l'équation différentielle avec condition initiale:

$$\frac{d}{dt}F(t) = \vec{0}, \quad F(0) = A$$

donc $t \mapsto exp(t\,X)(A)$ est une application constante de valeur égale à A. D'où le déplacement $\boldsymbol{R}_t = exp\,(t\,X) \in Rot(A)$ (où $Rot(A)$ est l'ensemble des rotations autour du point A). Cela étant pour tout point $A \in \Lambda_X$, donc $\boldsymbol{R} = exp\,X$ laisse invariant l'axe Λ_X. ∎

Théorème 2.20

Tout déplacement $\boldsymbol{D} \in \mathbb{D}$ se décompose en un déplacement de rotation $\boldsymbol{R} \in \mathbb{D}$ et un déplacement de translation $\boldsymbol{T} \in \mathbb{D}$ suivant un même axe, tels que:

$$\boldsymbol{D} = \boldsymbol{T}\,\boldsymbol{R} = \boldsymbol{R}\,\boldsymbol{T} \qquad \text{(Eq.2.27)}$$

Preuve :

Soit $\boldsymbol{D}$ un déplacement du groupe $\mathbb{D}$, notons X son générateur infinitésimal (i.e. $\boldsymbol{D} = exp\,X$).

- Si $X = 0$, alors, la translation et la rotation du théorème se réduisent à l'élément neutre e du groupe des déplacements.
- Si $X \in \mathfrak{t}$, alors, il suffit de prendre $\boldsymbol{T} = exp\,X$ et $\boldsymbol{R} = e$.
- Si $X \in \mathfrak{d} \setminus \mathfrak{t}$, alors, $\| X \|_{\mathfrak{d}}$ est inversible. Posons $Z = \dfrac{\mathsf{m}(X)}{\| X \|_{\mathfrak{d}}} X$ et $U = \dfrac{\mathsf{m}(X)\,\mathsf{p}(X)}{\| X \|_{\mathfrak{d}}} \Omega X$. Ainsi, les torseurs Z et U vérifient:
  - $U \in \mathfrak{t}$ et $Z \notin \mathfrak{t}$ tel que $[Z/Z] = 0$. Donc, d'après les propositions 2.17 et 2.19 le déplacement $\boldsymbol{T} = exp\,U$ est une translation et le déplacement $\boldsymbol{R} = exp\,Z$ est une rotation.
  - $X = Z + U$ et $[Z,U] = 0$, donc $exp\,U$ et $exp\,Z$ commutent (i.e. de même axe). Ainsi:

$$\boldsymbol{D} = exp\,X = (exp\,Z) \circ (exp\,U) = (exp\,U) \circ (exp\,Z)$$

Ce qui montre le théorème. ∎

[12] Voir le théorème 2.10 [page 40].

2.12 Changement de coordonnées relatif à un changement de configurations

Considérons une configuration de référence r d'un solide et une famille fondamentale $\mathfrak{f}_r = (\xi_r, \eta_r, \zeta_r)$ rigidement solidaire à r. A toute autre configuration s transformée de r par un déplacement D (i.e. $s = D \bullet r$), nous pouvons associer également une famille $\mathfrak{f}_s = (\xi_s, \eta_s, \zeta_s)$ telle que $\mathfrak{f}_s = D(t) \diamond \mathfrak{f}_r$. Les familles $\mathfrak{f}_r$ et $\mathfrak{f}_s$ en tant que bases ortho-normales, du Δ–module libre $\mathfrak{V}$ de dimension trois, définissent deux systèmes de coordonnées de $\mathfrak{V}$. Soit Y un torseur quelconque de $\mathfrak{V}$, notons Y_r et Y_s les vecteurs duaux colonnes (i.e matrices 3×1) qui représentent les coordonnées locales de Y, respectivement, dans les bases ortho-normales $\mathfrak{f}_r$ et $\mathfrak{f}_s$. Alors, en notant $[D_*]$ la matrice duale, associée à l'opérateur D_*, relativement à la base $\mathfrak{f}_r$, les systèmes de coordonnées du torseur Y par rapport à ces deux bases sont liées par la relation:

$$Y_{D \bullet r} = [D_*]\, Y_r \qquad \text{(Eq.2.28)}$$

Soit, maintenant, une transformation linéaire A dans $\mathfrak{V}$ (i.e. $A \in \mathcal{L}(\mathfrak{V})$). Si $[A]_r$ et $[A]_s$ représentent les matrices duales associées à A relativement aux familles fondamentales $\mathfrak{f}_r$ et $\mathfrak{f}_s$, alors:

$$[A]_{D \bullet r} = [D_*]\, [A]_r\, [D_*]^{-1} \qquad \text{(Eq.2.29)}$$

Cette relation traduit le changement de bases associé au changement de configuration du solide.

2.13 Conclusion

Famille fondamentale et configuration de corps rigide sont des représentations équivalentes à un isomorphisme près. Nous avons vu que la notion de famille fondamentale permet une "*algébrisation*" de la notion classique de repère affine orthonormé direct. En effet, il est difficile de munir un calcul direct des repères affines. Ce calcul se fait en deux étapes distincte et souvent fait appel à des techniques différentes:

- calcul de l'orientation du repère étudié dans un repère de référence,
- calcul de la position du repère étudié dans le repère de référence.

L'intérêt d'utiliser des familles fondamentales permet de traduire ces deux transformations en une seule transformation duale traduisant le déplacement associé au changement de configurations -changeant le repère associé à la famille fondamentale de référence en le repère associé à la famille fondamentale étudiée- de l'objet "*famille fondamentale*". Cette vision des choses permet de conduire un calcul symbolique à un degré d'abstraction assez élevé.

Partie 2:

MODELISATION DES SYSTEMES ARTICULES

– *Résumé de la partie 2* –

MODELES GEOMETRIQUE ET DYNAMIQUE DES MECANISMES

Cette partie présente l'ensemble des modèles de comportement des systèmes articulés à structure de chaîne cinématique ouverte simple ou arborescente dans le contexte théorique défini dans la première partie et basée sur la notion de structure algébrique qui va permettre d'opérer à un degré d'abstraction assez élevé. Les méthodes que nous y développons s'appliquent à une grande classe de systèmes mécaniques.

Soulignons que l'application de la théorie des groupes à l'ensemble des déplacements et à la caractérisation des liaisons mécaniques offre un langage naturel intéressant pour la description des modèles des mécanismes. Ce langage favorise en particulier une formulation analytique et un calcul symbolique simple (i.e. la syntaxe des modèles s'exprime de façon très simple dans un tel langage). Ces méthodes seront donc implémentées dans des processus de calcul formel. La théorie des graphes offrira alors l'outil permettant de généraliser ces modèles et méthodes à l'étude de systèmes articulés plus complexes.

Chapitre 3

MODELE GEOMETRIQUE DES SYSTEMES ARTICULES

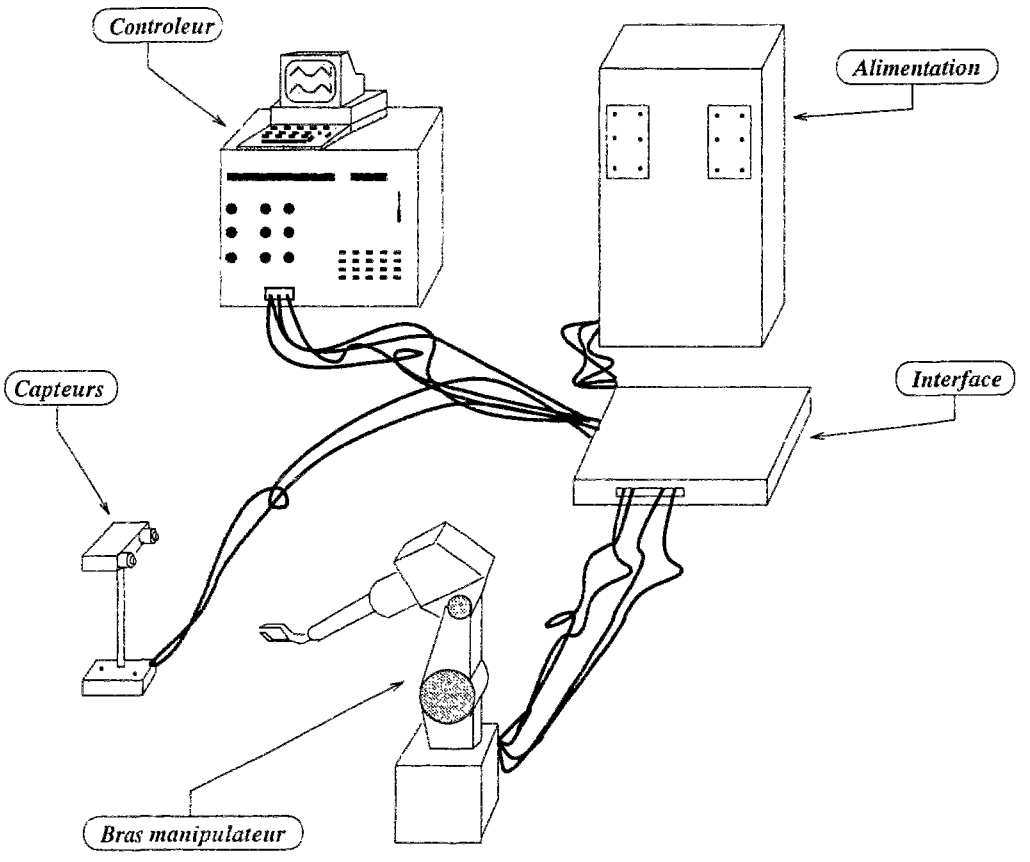
3.1 Introduction

Dans le domaine des applications robotiques, on décrit en général la tâche du manipulateur dans un repère lié à l'organe terminal. Le modèle géométrique du robot ou système articulé permet alors de traduire cette tâche dans le repère de travail (généralement lié au socle) [Fig. 3.1]. Dans ce chapitre, on considère des manipulateurs et des systèmes articulés sous forme de chaînes ouvertes (arborescentes ou simples) à $N+1$ éléments dont le corps de base est d'indice 0 (socle généralement fixe dans le cas de robots manipulateurs industriels). Les corps sont articulés par des liaisons mécaniques.

Les outils de modélisation que nous allons établir se basent sur les résultats de la partie 1 de ce mémoire. Nous présenterons d'abord une classification des paires cinématiques par les sous-groupes du groupe des déplacements $\mathbb{D}$ et la caractérisation des ensembles des déplacements cinématiquement admissibles par les articulations dans de telles paires cinématiques. Ensuite, nous allons donner quelques éléments de la théorie des graphes qui permettra d'étudier des systèmes à structure arborescente. La définition de cosinus directeurs duaux nous permettra de généraliser les conventions d'Euler et de Briant afin de pouvoir établir des paramétrages duaux définis par des cartes [Annexe A, paragraphe A.1.1, page 211] de la variété -de dimension six- des configurations du solide rigide (ou encore du groupe de Lie des déplacements, une fois une configuration de référence -du solide rigide- est préalablement fixée).

L'expression dans la théorie des groupes et algèbres de Lie du modèle géométrique des robots et systèmes articulés à structure de chaîne ouverte simple et la traduction des conventions de Denavit-Hartenberg, dans ce formalisme, en utilisant la notion des familles fondamentales permettent de mettre en évidence un langage mathématique propice à un calcul symbolique itératif du modèle

géométrique des chaînes cinématiques. L'utilisation de la théorie des graphes, nous permettra de



Cette figure schématise les différentes parties d'un système robotisé. L'interaction des différents éléments de ce système est assurée à travers un système d'interface qui acquiert les commandes du contrôleur et convertit les signaux en données numériques, le repère de travail étant celui des capteurs.

Fig. 3.1: Installation robotisée

généraliser le modèle géométrique itératif à des chaînes cinématique à structure arborescente.

3.2 Liaison – Paire cinématique – Degrés de liberté

Considérons un solide $\mathcal{C}$. Il est dit isolé s'il peut se mouvoir librement sans être soumis à aucune contrainte. Sinon, on dit qu'il est soumis à des liaisons. En mécanique, il existe plusieurs types de liaisons. Dans la littérature, les auteurs ont employé différents formalismes et différentes notations pour décrire de telles liaisons. Pour notre part, nous nous tiendrons à une terminologie en relation avec le formalisme des groupes[1].

En fait, le terme liaison a un sens purement mathématique. Cependant, il existe en général plusieurs moyens, en mécanique, de réaliser matériellement une liaison [Fig. 3.3]. Dans le cadre de cet exposé, nous allons modéliser une liaison d'origine mécanique par une relation binaire sur l'ensemble $\mathfrak{F}$ (des familles fondamentales). Considérons deux corps $\mathcal{C}_i$ et $\mathcal{C}_j$ entrant en liaison. Dans la suite nous allons adjoindre à ces deux corps deux familles fondamentales qui en soient rigidement solidaires[2]. Soient $\mathfrak{f}_i$ et $\mathfrak{f}_j$ ces deux familles; alors, à tout moment, on peut déterminer le mouvement relatif de $\mathcal{C}_j$ par rapport à $\mathcal{C}_i$ par un déplacement $\boldsymbol{D}(t)$ tel que: $\mathfrak{f}_j = \boldsymbol{D}(t) \diamond \mathfrak{f}_i$. Ce déplacement traduit la liaison $\mathcal{L}$ entre les corps $\mathcal{C}_i$ et $\mathcal{C}_j$ et l'on écrira:

$$\mathfrak{f}_i \, \mathcal{L} \, \mathfrak{f}_j \iff \mathfrak{f}_j = \boldsymbol{D}(t) \diamond \mathfrak{f}_i \qquad \text{(Eq.3.1)}$$

Définition 3.1

On appellera ensemble des déplacements cinématiquement admissibles pour la liaison $\mathcal{L}$, l'ensemble implicitement défini par:

$$\mathfrak{C}(i,j) = \{\boldsymbol{D} \in \mathbb{D} / \mathfrak{f}_j = \boldsymbol{D} \diamond \mathfrak{f}_i\} \qquad \text{(Eq.3.2)}$$

Remarquons que la détermination de l'ensemble des déplacements cinématiquement admissibles est obtenue à un déplacement (constant) près. En effet, considérons deux corps $\mathcal{C}_i$ et $\mathcal{C}_j$ et trois familles mobiles telles que par exemple:

- $\mathfrak{f}_i$ est rigidement solidaire au corps $\mathcal{C}_i$,
- $\mathfrak{f}_{j_1}$ et $\mathfrak{f}_{j_2}$ sont rigidement solidaires au corps $\mathcal{C}_j$.

Comme $\mathfrak{f}_{j_1}$ et $\mathfrak{f}_{j_2}$ sont rigidement solidaires d'un même corps, alors, il existe un déplacement fini -constant- $\boldsymbol{D}_2^1$ tel que:

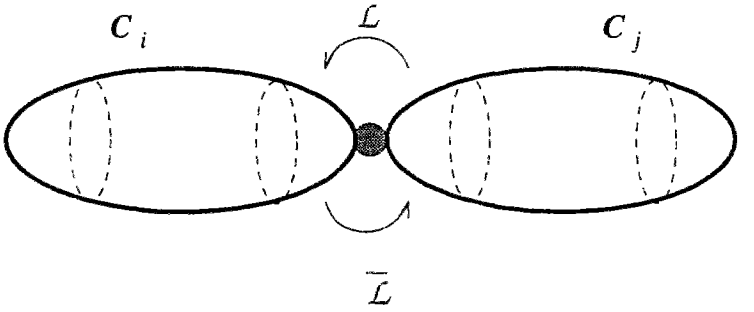
$$\mathfrak{f}_{j_2} = \boldsymbol{D}_2^1 \diamond \mathfrak{f}_{j_1}$$

Ainsi, les ensembles cinématiquement admissibles $\mathfrak{C}(i,j_1)$ et $\mathfrak{C}(i,j_2)$ sont liés par la relation:

$$\mathfrak{C}(i,j_2) = R_{D_2^1} \, \mathfrak{C}(i,j_1) \qquad \text{(Eq.3.3)}$$

[1] Le formalisme que nous adopterons pour la modélisation des liaisons s'inspire des travaux de D. Chevallier [CHEV-86] et M. Hervé [HERV-78].

[2] Cela signifie que lorsque ces deux corps sont animés de mouvement les deux familles se comporteront comme deux familles mobiles chacune animée du même mouvement que le corps auquel elle est associée.



A la liaison $\mathcal{L}$, entre les corps $\mathcal{C}_i$ et $\mathcal{C}_j$, correspond une liaison conjuguée $\overline{\mathcal{L}}$ telle que: $\mathfrak{f}_i\,\mathcal{L}\,\mathfrak{f}_j \rightarrow \mathfrak{f}_j\,\mathcal{L}\,\mathfrak{f}_i$. Ainsi, pour toute liaison entre deux corps $\mathcal{C}_i$ et $\mathcal{C}_j$: $\overline{\overline{\mathcal{L}}} \equiv \mathcal{L}$.

Fig. 3.2: Liaison entre deux corps

où $R_{D_2^1}$ désigne l'action de multiplier, à droite[3], tous les déplacements de l'ensemble $\mathfrak{C}(i,j_1)$ par le déplacement $\boldsymbol{D}_2^1$.

Par ailleurs, La réciprocité du mouvement relatif entre les corps $\mathcal{C}_i$ et $\mathcal{C}_j$ entraîne qu'à la liaison $\mathcal{L}$ reliant le corps $\mathcal{C}_j$ par référence au corps $\mathcal{C}_i$ est associée une liaison $\overline{\mathcal{L}}$ reliant le corps $\mathcal{C}_i$ par référence au corps $\mathcal{C}_j$, telle que:

$$\mathfrak{f}_j\,\overline{\mathcal{L}}\,\mathfrak{f}_i \iff \mathfrak{f}_i\,\mathcal{L}\,\mathfrak{f}_j \qquad \text{(Eq.3.4)}$$

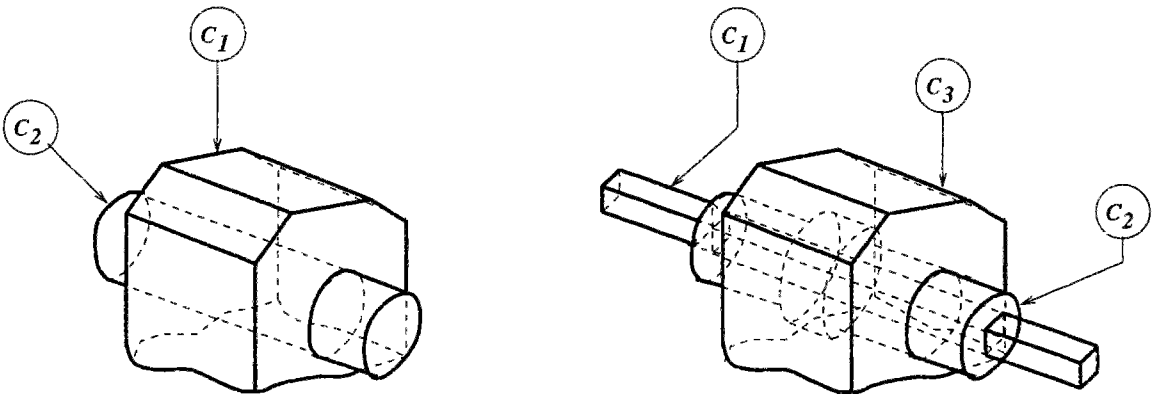
L'ensemble des déplacements cinématiquement admissibles $\overline{\mathfrak{C}}(j,i)$ associé à la liaison $\overline{\mathcal{L}}$ est donné par la relation:

$$\overline{\mathfrak{C}}(j,i) = \quad \mathfrak{C}^{-1}(i,j) \qquad \text{(Eq.3.5)}$$

où $\mathfrak{C}^{-1}(i,j)$ désigne l'ensemble des inverses des déplacements de l'ensemble $\mathfrak{C}(i,j)$. On a immédiatement:

[3]Explicitement, cela signifie:

$$R_{D_2^1}\,D = \; D\,D_2^1\,.$$



(a) (b)

Dans cet exemple, on matérialise la liaison cylindrique de deux façons différentes:

(a) La liaison cylindrique est réalisée à partir de la paire $\mathbb{S}_{1\,2} = \{\mathcal{C}_1, \mathcal{C}_2\}$ (verrou).

(b) La liaison cylindrique est réalisée à partir de la paire $\mathbb{S}_{1\,3} = \{\mathcal{C}_1, \mathcal{C}_3\} = \{\mathcal{C}_1, \mathcal{C}_2\} \circ \{\mathcal{C}_2, \mathcal{C}_3\}$ par composition des liaisons de la paire $\mathbb{S}_{12} = \{\mathcal{C}_1, \mathcal{C}_2\}$ (paire prismatique) et de la paire $\mathbb{S}_{2\,3} = \{\mathcal{C}_1, \mathcal{C}_3\}$ (paire rotoïde).

Dans l'exemple (a) le montage est réalisé à partir de deux pièces mécaniques, alors que dans l'exemple (b) le montage est réaliser à partir de trois pièces mécaniques. Cependant, les deux montages définissent la même liaison mécanique.

Fig. 3.3: Matérialisation d'une paire cinématique

Proposition 3.1

Si $\mathfrak{C}(i,j)$ est un sous-groupe de $\mathbb{D}$, alors:

$$\overline{\mathfrak{C}}(j,i) = \mathfrak{C}(i,j) \qquad \text{(Eq.3.6)}$$

Définition 3.2

- *Une liaison géométrique (ou "holonome[4]") entre deux corps $\mathcal{C}_i$ et $\mathcal{C}_j$ est une relation binaire $\mathcal{L}$ dans $\mathfrak{F}$ qui vérifie les deux axiomes suivants:*

  (h_1) *Si $\mathfrak{f}_i$ et $\mathfrak{f}_j$ sont deux familles fondamentales rigidement solidaires respectivement des corps $\mathcal{C}_i$ et $\mathcal{C}_j$:* $\qquad \forall \boldsymbol{D} \in \mathbb{D} : \quad \mathfrak{f}_i \,\mathcal{L}\, \mathfrak{f}_j \implies \boldsymbol{D} \diamond \mathfrak{f}_i \,\mathcal{L}\, \boldsymbol{D} \diamond \mathfrak{f}_j$

  (h_2) *L'ensemble $\mathfrak{C}(i,j)$ des déplacements cinématiquement admissibles par la liaison $\mathcal{L}$ est une sous-variété différentielle de $\mathbb{D}$.*

[4]Il est vrai qu'on mécanique on a l'habitude d'appeler liaison holonome une liaison définie par une relation mathématique qui fait intervenir les positions des solides mais pas les dérivées, par rapport au temps, des positions. Cependant, toutes les liaisons holonomes qu'on sait réaliser matériellement vérifient la définition qu'on adopte dans cet exposé (les deux axiomes (h_1) et (h_2)).

- *Dans le cas contraire, la liaison sera dite cinématique (ou "non-holonome").*

Un grand nombre de liaisons mécaniques matérialisent des sous-groupes de $\mathbb{D}$ [Tableau 3.1, page 67]. Elles vérifient en fait un axiome plus fort que l'axiome (h_2):

(h_2') L'ensemble $\mathfrak{C}(i,j)$ des déplacements cinématiquement admissibles par la liaison $\mathcal{L}$ est un sous-groupe de Lie du groupe $\mathbb{D}$.

Par ailleurs, l'axiome (h_1) signifie que si les familles $\mathfrak{f}_i$ et $\mathfrak{f}_j$ sont congrues modulo la relation $\mathcal{L}$, alors, il en est de même pour les famille $\boldsymbol{D}\diamond\mathfrak{f}_i$ et $\boldsymbol{D}\diamond\mathfrak{f}_j$ pour tout déplacement $\boldsymbol{D}\in\mathbb{D}$ [Fig. 3.4]. Cela va traduire le fait que dans un système multicorps, le déplacement qu'induit un changement de configurations d'un élément de référence, sur les configurations des autres éléments de la chaîne articulée, n'affecte pas la nature de la liaison entre ces corps.

Définition 3.3

- *Une paire cinématique est la réalisation matérielle -technologique- d'une liaison (géométrique en général) entre deux corps.*
- *Une paire cinématique inférieure est une paire cinématique qui réalise un sous-groupe du groupe des déplacements (par des contacts surfaciques, selon des surfaces invariantes par les sous-groupes).*
- *Une paire cinématique supérieure est une paire cinématique qui correspond à un contact ponctuel ou linéaire [Fig. 3.5 & Fig. 3.6].*

Définition 3.4

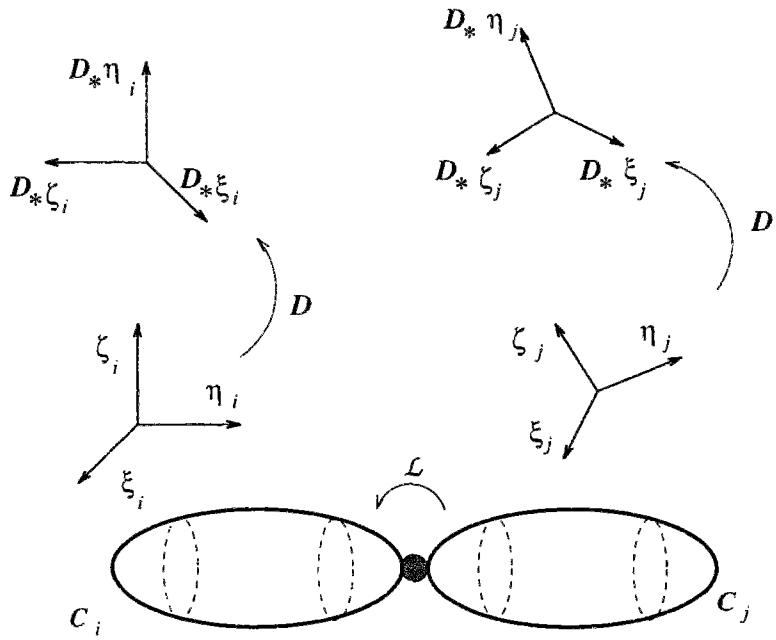
Le nombre de degrés de liberté d'un solide en mouvement est le nombre de paramètres indépendants nécessaires à la description de la variété des configurations du solide dans son mouvement.

La notion de groupe à un paramètre joue un rôle essentiel dans la modélisation de liaisons, puisque les déplacements cinématiquement admissibles d'une liaison géométrique quelle qu'elle soit, peuvent être décomposés en un produit de sous-groupes à un paramètre[5]. En effet, nous avons vu que dans une configuration donnée, on peut adjoindre au corps $\mathcal{C}$ une famille fondamentale $\mathfrak{f} = (\xi,\eta,\zeta)$ à laquelle est associé un repère orthonormé $\mathcal{R}$ (matérialisé par les axes centraux des éléments de la famille $\mathfrak{f}$). Ainsi, si une famille fondamentale $\mathfrak{f} = (\xi,\eta,\zeta)$ est fixée et si une configuration de référence de $\mathcal{C}$ est choisie, alors il existe un unique isomorphisme de $\mathfrak{F}$ sur $\mathcal{S}$. Par la suite, le nombre de degrés de liberté permis par une liaison peut être défini comme étant le nombre de sous-groupes à un paramètre indépendants nécessaires pour engendrer la variété différentielle des déplacements cinématiquement admissibles par cette liaison.

Exemple

Ainsi, une façon de compter ces sous-groupes à un paramètre pour un solide isolé $\mathcal{C}$ consiste à dire que l'ensemble des déplacements cinématiquement admissibles est le groupe $\mathbb{D}$ en entier, donc $\mathcal{C}$ peut effectuer:

[5] Cette idée se réfère à la notion de coordonnée de seconde espèce de la paramétrisation d'un groupe de Lie [Annexe A, paragraphe A.5.5, page 220].



Cette figure schématise une interprétation géométrique de l'axiome (h_1):

$$\mathfrak{f}_i \mathcal{L} \mathfrak{f}_j \implies D \diamond \mathfrak{f}_i \mathcal{L} D \diamond \mathfrak{f}_j, \quad \forall D \in \mathbb{D}$$

Elle exprime le fait qu'une liaison géométrique implique que si deux famille fondamentales (i.e les corps auxquels elles sont liées) sont congrues modulo une liaison, alors, la propriété reste vrai à un déplacement près de tout le bloc $\{\mathfrak{f}_i, \mathfrak{f}_j\}$.

Fig. 3.4: Interprétation géométrique

- trois translations indépendantes suivant les axes $\Lambda_\xi, \Lambda_\eta$ et Λ_ζ, à savoir:

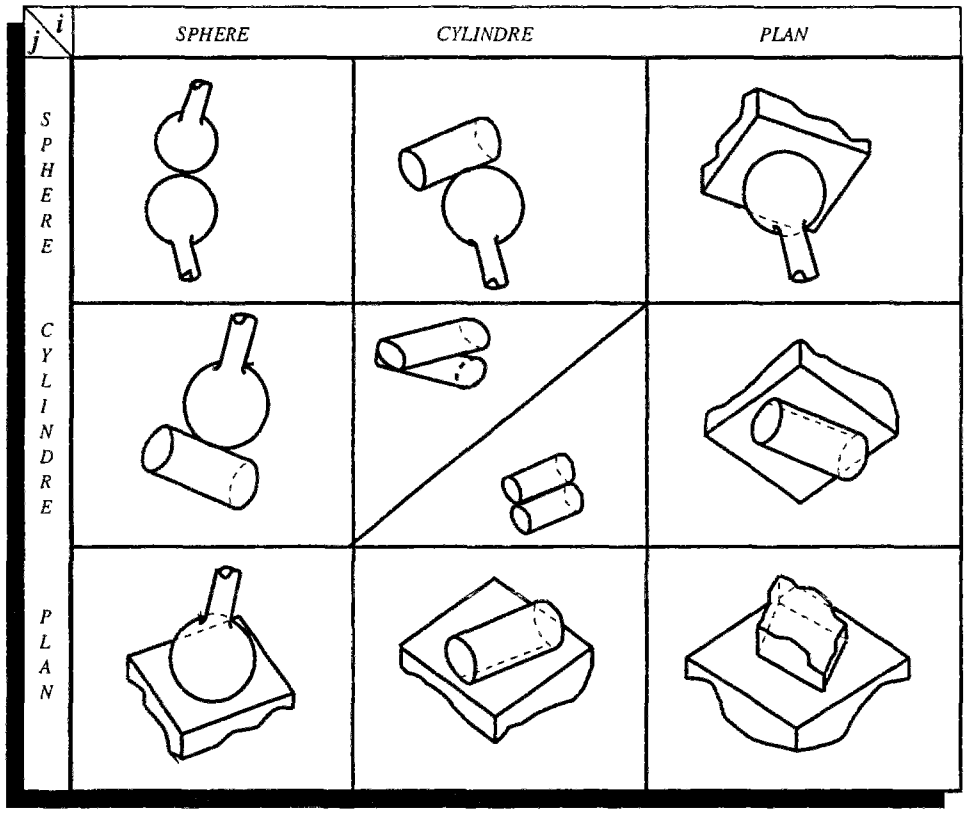
$$\boldsymbol{T}_\xi = exp(a\,\Omega\,\xi), \quad \boldsymbol{T}_\eta = exp(b\,\Omega\,\eta) \quad \text{et} \quad \boldsymbol{T}_\zeta = exp(c\,\Omega\,\zeta). \qquad \text{(Eq.3.7)}$$

où a, b et c sont les longueurs respectives des translations $\boldsymbol{T}_\xi$, $\boldsymbol{T}_\eta$ et $\boldsymbol{T}_\zeta$.

- trois rotations autour des axes $\Lambda_\xi, \Lambda_\eta$ et Λ_ζ, soit:

$$\boldsymbol{R}_\xi = exp(\alpha\,\xi), \quad \boldsymbol{R}_\eta = exp(\beta\,\eta) \quad \text{et} \quad \boldsymbol{R}_\zeta = exp(\gamma\,\zeta). \qquad \text{(Eq.3.8)}$$

où α, β et γ sont les angles respectifs des rotations $\boldsymbol{R}_\xi$, $\boldsymbol{R}_\eta$ et $\boldsymbol{R}_\zeta$.



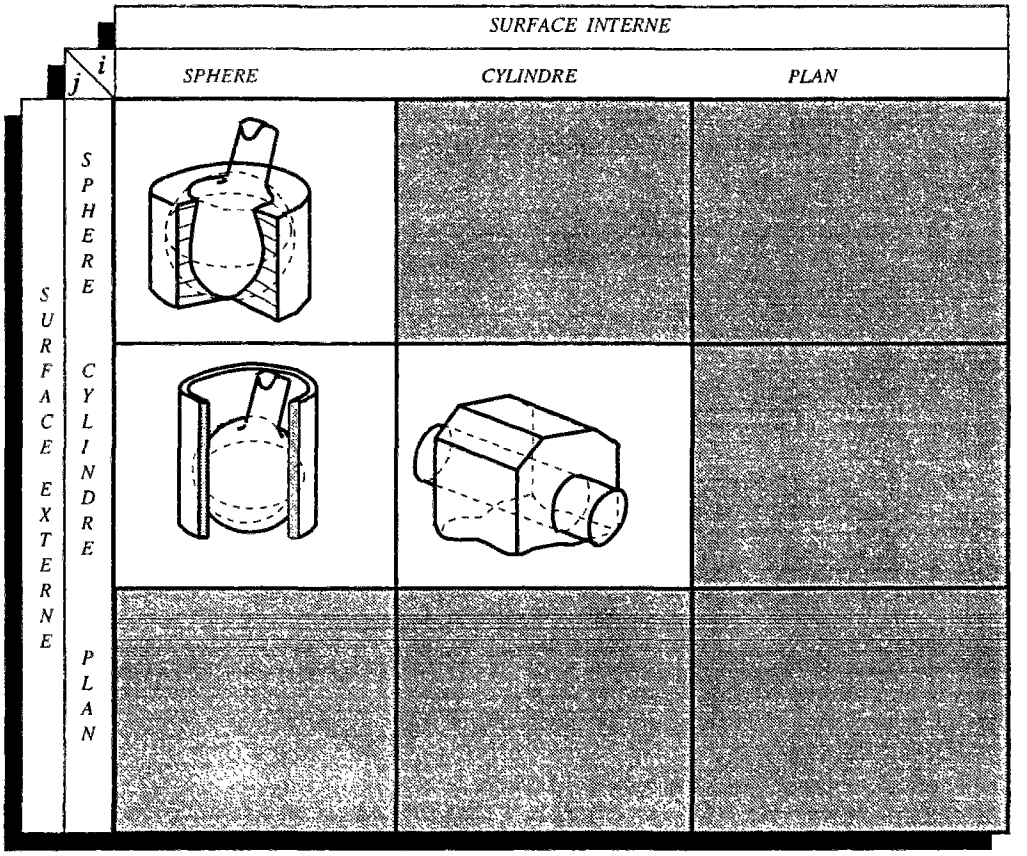
A l'exception du contact plan–plan, dans un contact externe entre deux surfaces usuelles obtenues par usinage d'un matériau, tous les autres contacts définissent des paires cinématiques supérieur.

Fig. 3.5: Paires cinématiques réalisées par contact entre surfaces externes

Un solide isolé a donc six degrés de liberté (d.d.l.). Dès qu'une liaison est établie entre deux solides, chacun d'eux perd de ses d.d.l. vis-à-vis du second.

3.3 Classification des paires cinématiques par sous-groupes de $\mathbb{D}$

La théorie des groupes offre une méthode très simple pour la classification des paires cinéma-

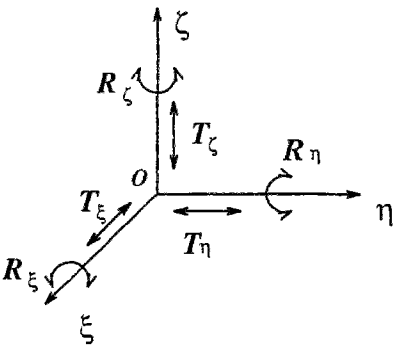


Les seuls assemblages surfaciques de type surface interne contre surface externe possibles (entre surfaces élémentaires: sphère, cylindre, plan) sont:

- L'assemblage sphère intérieure contre sphère circonscrite extérieure qui définit une paire inférieure.
- L'assemblage cylindre intérieur contre cylindre circonscrit extérieur qui définit également une paire inférieure.
- L'assemblage sphère intérieure contre cylindre circonscrit extérieur qui, quant à lui, définit une paire supérieure.

Fig. 3.6: Paires cinématiques réalisées par contact interne–externe

tiques. En effet, quelle que soit la réalisation matérielle d'une liaison mécanique, les déplacements cinématiquement admissibles pour cette liaison peuvent être décomposés en un produit de sous-groupes de $\mathbb{D}$. Grâce à l'application exponentielle définie dans le contexte de théorie des groupes de Lie et la notion de familles fondamentales, il est possible de caractériser explicitement ces déplacements pour permettre de faire un calcul symbolique direct dans l'algèbre de Lie $\mathfrak{d}$ (par exemple:



Cette figure schématise les sous-groupes à un paramètre associés aux six degrés de liberté relativement aux axes d'une famille fondamentale.

Fig. 3.7: Degrés de liberté

la transformation de familles fondamentales).

Dans le tableau 3.1, les déplacements cinématiquement admissibles sont donnés dans le système coordonnées de seconde espèce $(\alpha, a, \beta, b, \gamma, c)$ associées à la base $(\xi, \Omega\,\xi, \eta, \Omega\,\eta, \zeta, \Omega\,\zeta)$ de $\mathfrak{d}$ liée à la famille fondamentale $\mathfrak{f} = (\xi, \eta, \zeta)$ convenablement choisie [Annexe A, paragraphe A.5.5, page 220]. Ces déplacements peuvent avoir des expressions différentes dans d'autres systèmes de coordonnées. La correspondance entre les différentes représentations sont obtenues par des changements de cartes sur le groupe de Lie $\mathbb{D}$.

3.4 Chaîne cinématique

Un solide peut éventuellement entrer en liaison dans la réalisation technologique de plusieurs paires cinématiques à la fois. Cette aspect de la mécanique donne lieu à la notion de chaîne cinématique ou mécanisme.

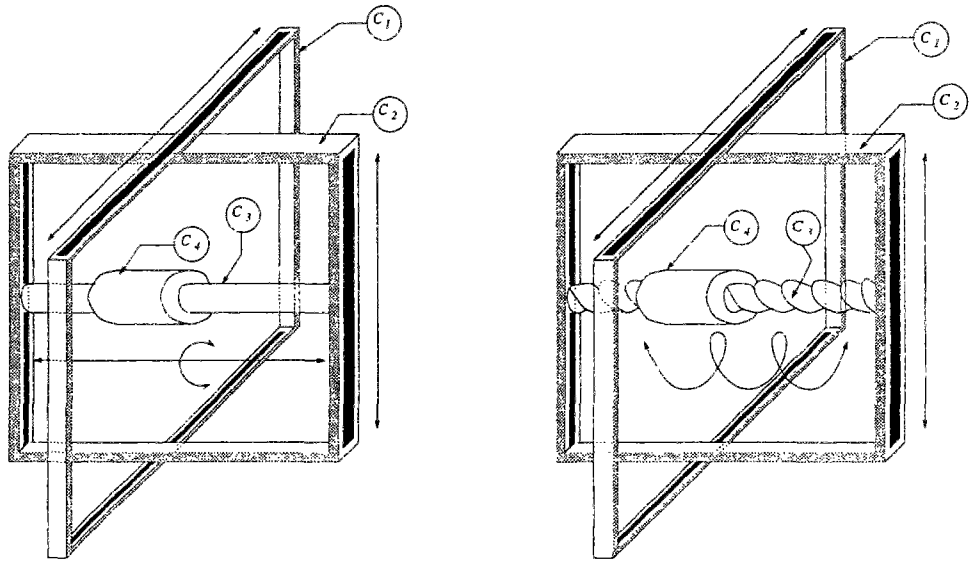
Définition 3.5

Une chaîne cinématique (i.e. mécanisme) est la réalisation matérielle, technologique, d'un assemblage de N corps (rigides), tels que chacun de ses éléments entre en liaison (géométrique en général) avec au moins un deuxième élément de cet assemblage. On distingue deux grandes classes de chaînes cinématiques: les chaînes cinématiques simples et les chaînes cinématiques complexes.

Tableau 3.1 : *Classification des paires cinématiques par sous-groupes de* $\mathbb{D}$

Paire	*Type*	*Caractérisation géométrique de la liaison*	*Déplacements cinématiquement admissibles*
Encastrée	E_0 (0 d.d.l.)	n'autorise pas de mouvements relatifs.	e Elément neutre du groupe $\mathbb{D}$.
Prismatique	P_ζ (1 d.d.l.)	le contact prend effet sur une surface prismatique. Les déplacements cinématiquement admissibles s'obtiennent par des translations sur le bord du prisme de contact.	$exp(c\,\varepsilon\,\zeta)$ tel que ζ est un glisseur unitaire et l'axe Λ_ζ matérialise le bord du prisme.
Pivot	R_ζ (1 d.d.l.)	le contact est établi sur une surface cylindrique. Les déplacements cinématiquement admissibles, par une telle liaison, s'obtiennent par une rotation autour de l'axe du cylindre de contact.	$exp(\gamma\,\zeta)$ tel que ζ est un glisseur unitaire et Λ_ζ matérialise l'axe de rotation.
Hélicoïdale	$H_{p,\zeta}$ (1 d.d.l.)	le contact est défini par une surface de la forme d'une vis. Cette surface est entièrement déterminée par la donné de l'axe du cylindre inscrit et le pas p définie par la vis.	$exp(\gamma\,(1+p\,\varepsilon)\,\zeta)$ tel que ζ est un glisseur unitaire et Λ_ζ matérialise l'axe de la vis.
Cylindrique	C_ζ (2 d.d.l.)	la surface de contact entre ses deux éléments est un cylindre de section circulaire. Les déplacements cinématiquement admissibles s'obtiennent par la composée d'une rotation dont l'axe est celui du cylindre et une translation le long du même axe.	$exp((\gamma+c\,\varepsilon)\,\zeta)$ tel que ζ est un glisseur unitaire et Λ_ζ matérialise l'axe du cylindre.
Translation plane	$P_{2\zeta}$ (2 d.d.l.)	les éléments d'une telle paire peuvent translater librement l'un par rapport à l'autre en restant dans un même plan.	$exp(\varepsilon\,(a\,\xi+b\,\eta))$ tel que $\mathfrak{f} = (\xi,\eta,\zeta)$ forme une famille fondamentale.
Plane	Π_ζ (3 d.d.l.)	en plus des translations dans le plan de contact, les éléments de la paire peuvent subir des rotations autour d'une normale à ce plan.	$exp(\varepsilon\,(a\,\xi+b\,\eta))\;exp(\gamma\,\zeta)$ tel que $\mathfrak{f} = (\xi,\eta,\zeta)$ forme une famille fondamentale.
Translation spatiale	P_3 (3 d.d.l.)	la liaison permet des translations indépendantes, des éléments de la paire l'un par rapport à l'autre, dans les trois directions de l'espace.	$exp(\varepsilon\,(a\,\xi+b\,\eta+c\,\zeta))$ tel que $\mathfrak{f} = (\xi,\eta,\zeta)$ forme une famille fondamentale.
Sphérique	S_O (3 d.d.l.)	le contact entre les éléments de la paire prend lieu sur une sphère. La situation de chacun des éléments de la paire est donnée par rapport à celle de son partenaire par trois angles de rotation autour du centre O de la sphère de contact.	$exp(\alpha\,\xi)\;exp(\beta\,\eta)\;exp(\gamma\,\zeta)$ tel que $\mathfrak{f} = (\xi,\eta,\zeta)$ forme une famille fondamentale d'origine le point O.
Y	$Y_{p,\zeta}$ (3 d.d.l.)	définie par des déplacements hélicoïdaux suivant une direction de l'espace et des translations cardans indépendantes dans deux directions du plan perpendiculaire à cet axe [Fig. 3.8].	$exp(\varepsilon\,(\alpha\,\xi+\beta\,\eta))\;exp(\gamma\,(1+\varepsilon\,p)\,\zeta)$ tel que $\mathfrak{f} = (\xi,\eta,\zeta)$ forme une famille fondamentale.
X	X_ζ (4 d.d.l.)	définie par des déplacements cylindriques suivant une direction de l'espace et des translations cardans indépendantes dans deux directions du plan perpendiculaire à l'axe des déplacements cylindriques [Fig. 3.8].	$exp(\varepsilon\,(\alpha\,\xi+\beta\,\eta))\;exp((\gamma+\varepsilon c)\,\zeta)$ tel que $\mathfrak{f} = (\xi,\eta,\zeta)$ forme une famille fondamentale.
Libre	E (6 d.d.l.)	n'impose aucune contrainte	$exp((\alpha+\varepsilon a)\,\xi)\;exp((\beta+\varepsilon b)\,\eta)\;exp((\gamma+\varepsilon c)\,\zeta)$ tel que $\mathfrak{f} = (\xi,\eta,\zeta)$ forme une famille fondamentale.

- *Une chaîne cinématique est dite simple, si chacun de ses éléments participe à la réalisation d'au plus deux paires cinématiques (de la chaîne cinématique en question). Il existe deux*



Cette figure schématise deux mécanismes cardans conçu sur la base de la classification algébrique des mouvements que nous décrivons au tableau 3.1:

- Dans le mécanisme de droite:
  - Les paires $\{C_1, C_2\}$ et $\{C_2, C_3\}$ réalisent des mouvements prismatiques cardans.
  - La paire $\{C_3, C_4\}$ réalise un mouvement hélicoïdal.
  - La paire $\{C_1, C_4\}$ réalise un mouvement Y.
- Dans le mécanisme de gauche:
  - Les paires $\{C_1, C_2\}$ et $\{C_2, C_3\}$ réalisent des mouvements prismatiques cardans.
  - La paire $\{C_3, C_4\}$ réalise un mouvement cylindrique.
  - La paire $\{C_1, C_4\}$ réalise un mouvement X.

Les mouvements Y et X, dans les deux cas de figure, sont obtenus au terme de la composition des mouvements des paires $\{C_1, C_2\}$, $\{C_2, C_3\}$ et $\{C_3, C_4\}$ pour la loi de composition interne du groupe $\mathbb{D}$.

Fig. 3.8: Exemples de mécanismes et de paires cinématiques

types de chaînes cinématiques simples:

- *une chaîne cinématique simple est dite ouverte si elle possède deux corps extrêmes (qui entrent dans la réalisation d'une paire cinématique et une seulement),*
- *sinon, la chaîne cinématique est dite fermée (ou encore une boucle).*

- *Si une chaîne cinématique n'est pas simple, alors, elle est dite complexe (ou composée). Là encore on distingue deux types de chaînes cinématiques complexes:*

- *si une chaîne cinématique complexe ne possède pas de boucles, elle est dite arborescente,*
- *sinon, elle est dite bouclée.*

3.5 Structure topologique des mécanismes

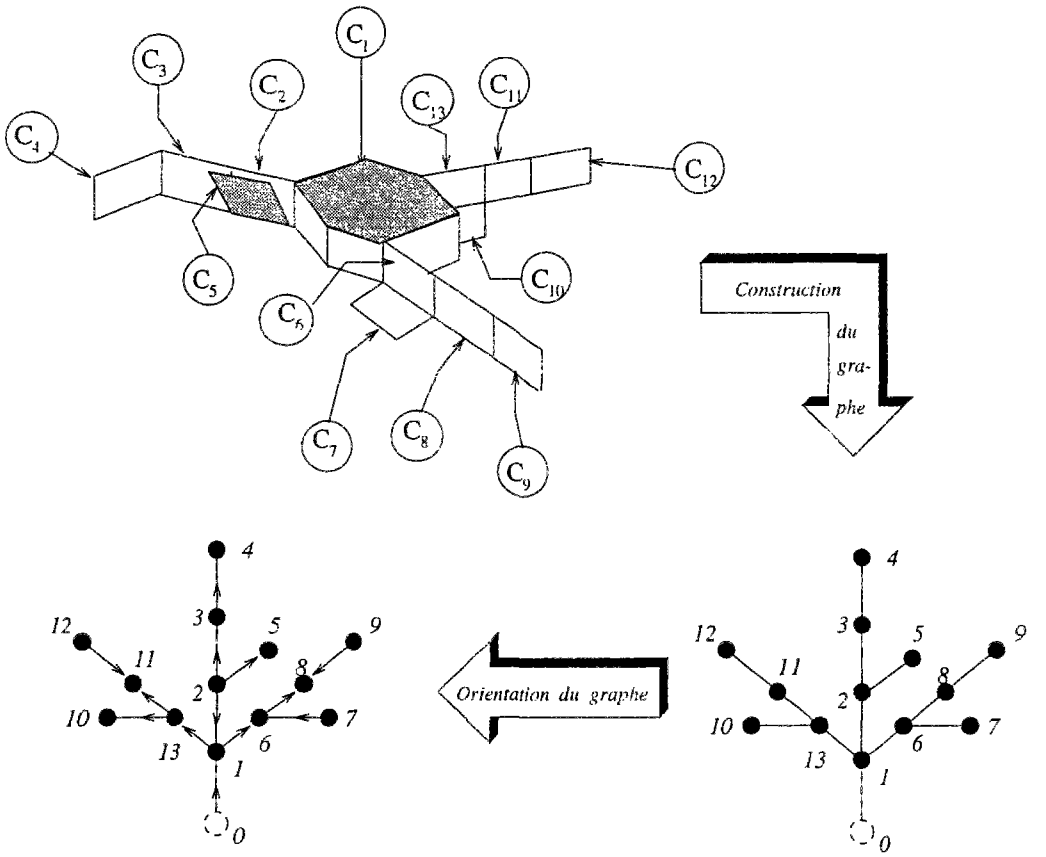
Nous entendons par topologie, les propriétés morphologiques invariantes d'une structure articulée indépendamment de la nature des liaisons qui définissent ses articulations. A toute structure topologique, on peut associer un graphe, tel que les arcs de ce graphe représentent les articulations et les sommets du graphe représentes les éléments du système articulé. L'objet de cette section est de caractériser le graphe des chaînes cinématiques arborescentes.

3.5.1 Graphe des mécanismes à topologie arborescente

Dans ce paragraphe, on étudie la structure topologique des mécanismes à chaînes cinématiques arborescentes. Un mécanisme à structure arborescente et N degrés de liberté est constitué par $N+1$ corps dont un élément de référence d'indice 0. Cet élément sera dit racine de la structure arborescente (Exemples: dans le cas des robots manipulateurs, l'élément d'indice 0 sera le socle, généralement fixe. Dans le cas de la modélisation des satellites et vaisseaux à structure arborescente, on pourra prendre un corps fictif $\mathcal{C}_0$ lié à l'espace), N articulations et K organes terminaux. En général, on peut numéroter arbitrairement de 1 à N les éléments autres que le corps $\mathcal{C}_0$. Pour numéroter les articulations, on attribue le même indice à un corps et son articulation amont. La construction du graphe d'un tel mécanisme consiste à [Fig. 3.10]:

- la donnée d'une racine du graphe (en général cela représentera l'élément $\mathcal{C}_0$),
- la donnée d'un ensemble de N sommets représentant les éléments de la structure articulée,
- la donnée d'un ensemble de N segments joignant les sommets, tels que chaque segment correspond à une articulation entre deux éléments du mécanisme.

Le graphe d'une chaîne arborescente est donc un arbre. Maintenant, pour obtenir le graphe orienté de la structure arborescente, il suffit d'attribuer un sens d'orientation, arbitraire à priori, aux arêtes du graphe ainsi obtenu. De tels segments orientés sont dits des arcs. Une fois qu'on a construit le graphe orienté de la structure arborescente, on peut déterminer les matrices d'incidence et de référence d'un tel graphe qui permettent l'automatisation du processus de description de la topologie de la structure arborescente [J. Wittenburg [WITT-77]]. L'approche que nous adopterons est différente. En fait, nous travaillons simplement sur des arbres et par la suite nous pouvons introduire la notion de graphe d'arbres ordonnés. En effet, nous allons introduire un ordre naturel pour le parcours des arbres. De ce fait, nous n'aurons plus besoin de la détermination des matrice d'incidence et de référence du graphe orienté (en entier). Cependant, nous aurons besoin de connaître les prédécesseurs immédiats des éléments de l'arbre. Bien évidement, dans les problèmes (d'automatisation du processus de modélisation) qui nous intéressent, le caractère arbitraire, du choix d'indiçage des sommets et l'orientation des arcs du graphe associé à la structure articulée,



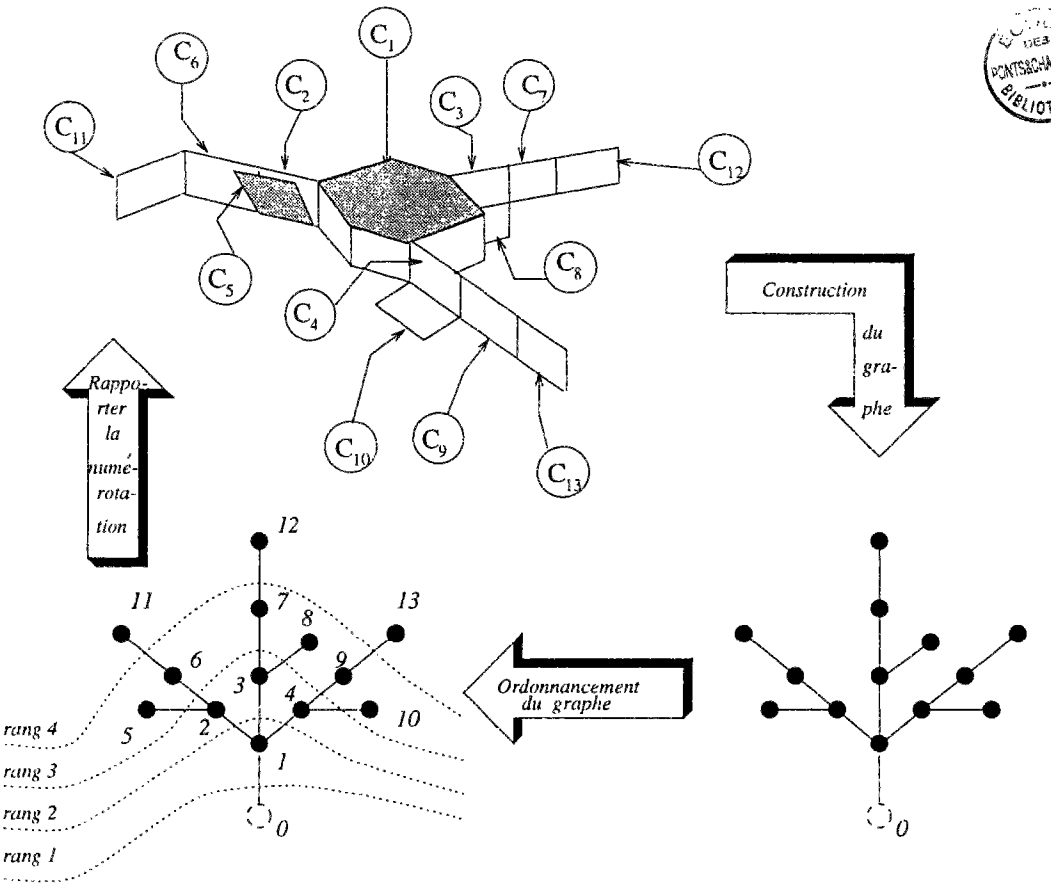
Dans une première étape, on commence par numéroter les éléments du satellite. Ensuite, après le choix d'un élément de référence, on peut dessiner le graphe associée à la structure arborescente. Enfin, pour obtenir le graphe orienté, on donne aux arcs des orientation arbitraire à priori.

Fig. 3.9: Graphe associé à une structure arborescente (graphe orienté)

augmente inutilement la complexité du problème et n'arrange en rien le caractère itératif que l'on souhaite établir pour mener les calculs (des modèles géométriques et dynamiques des mécanismes à structure arborescente).

3.5.2 Parcours en largeur de l'arbre topologique

Pour construire un ordre récursivement reconnaissable sur un arbre, il faut tout d'abord choisir un élément de référence (racine de l'arbre: corps éventuellement fictif) d'indice 0. Ensuite, pour pouvoir faire correctement la récurrence sur les indices des corps, les solides composant la



Dans une première étape, on commence par dessiner le graphe de la structure. On classifie les nœuds de l'arbre selon leurs profondeurs. On numérote les nœuds de l'arbre. Enfin, on rapporte cette numérotation aux éléments correspendant du système articulé. C'est cette méthode qui sera retenue dans la suite de notre discours.

Fig. 3.10: Graphe associé à une structure arborescente (graphe ordonné)

structure articulée doivent nécessairement être indicés de 1 à N de façon que le parcours -sans retour en arrière- des indices des sommets de l'arbre topologique associé à la structure du système articulé, se fasse dans un ordre croissant du socle vers les différents organes terminaux. Enfin, la numérotation des articulations doit pouvoir être déduite de celle des éléments du mécanisme.

Définition 3.6

Nous appellerons profondeur ou rang d'un élément, le nombre des liaisons amont reliant ce corps à l'élément d'indice 0. Un ordre sera dit "de parcours en largeur" du graphe associé à un

mécanisme à structure arborescente, s'il obéit aux règles suivantes:

- *On se fixe un élément d'indice 0 (éventuellement fictif).*
- *On trace le graphe associé au système articulé.*
- *On numérote dans l'ordre croissant les éléments du premier rang, en partant de l'élément de l'extrême gauche de la représentation graphique du robot (ou du mécanisme) à structure arborescente (cette règle est conventionnelle puisque il y a plusieurs façons de représenter l'arbre topologique associé à un mécanisme à structure arborescente).*
- *On numérote de la même manière les éléments du second rang, en commençant d'abord par les successeurs du corps d'indice 1 et ainsi de suite.*
- *On réitère le même processus aux éléments du troisième rang et ainsi de suite jusqu'à la numérotation de tous les organes terminaux.*
- *La numérotation des liaison sera déduite de celle des éléments du robot (ou mécanisme), en affectant le même indice à l'élément et sa liaison amont.*

Ainsi, on peut construire la matrice $\Upsilon = (u_{ij})_{i,j=0..N}$ des successeurs immédiats des éléments de la structure arborescente, d'ordre $(N+1) \times (N+1)$, telle que:

$$u_{ij} = \begin{cases} 1 & \text{si } \mathcal{C}_j \text{ est directement lié à } \mathcal{C}_i \text{ et } i < j \\ 0 & \text{sinon} \end{cases} \qquad \text{(Eq.3.9)}$$

où $\mathcal{C}_k$ désigne l'élément d'indice k ($k \in [0..N]$) du système articulé. Ainsi, si $u_{ij} = 1$, cela signifie que le corps $\mathcal{C}_j$ est un successeur immédiat du corps $\mathcal{C}_i$. A partir de la matrice Υ, on peut donc construire deux applications i^+ et i^- qui donnent respectivement l'ensemble des successeurs immédiats et le prédécesseur immédiat des sommets du graphe ordonné.

Définition 3.7

Soit N le nombre des articulations[6] d'un mécanisme à structure arborescente ouverte. Considérons I_N l'ensemble fini des $N+1$ premiers entiers naturels (i.e. $I_N = \{0, 1, \cdots, N\}$), I_N^* l'ensemble I_N privé de l'élément 0 et $\mathcal{P}(I_N^*)$ l'ensemble des parties de I_N^*:

- *On appellera indicatrice des successeurs immédiats des éléments de la structure arborescente; l'application $i^+ : I_N \rightarrow \mathcal{P}(I_N^*)$ qui à un entier k de I_N associe l'ensemble des indices des successeurs immédiats de l'élément $\mathcal{C}_k$ dans l'arbre topologique du robot ou système articulé.*
- *On appellera indicatrice du prédécesseur immédiat d'un élément du robot; l'application $i^- : I_N^* \rightarrow I_N$ qui à un entier k de I_N^* associe l'indice du prédécesseur immédiat de l'élément $\mathcal{C}_k$ dans l'arborescence.*

De cette définition et de celle de la matrice Υ, il découle immédiatement la proposition suivante:

Proposition 3.2

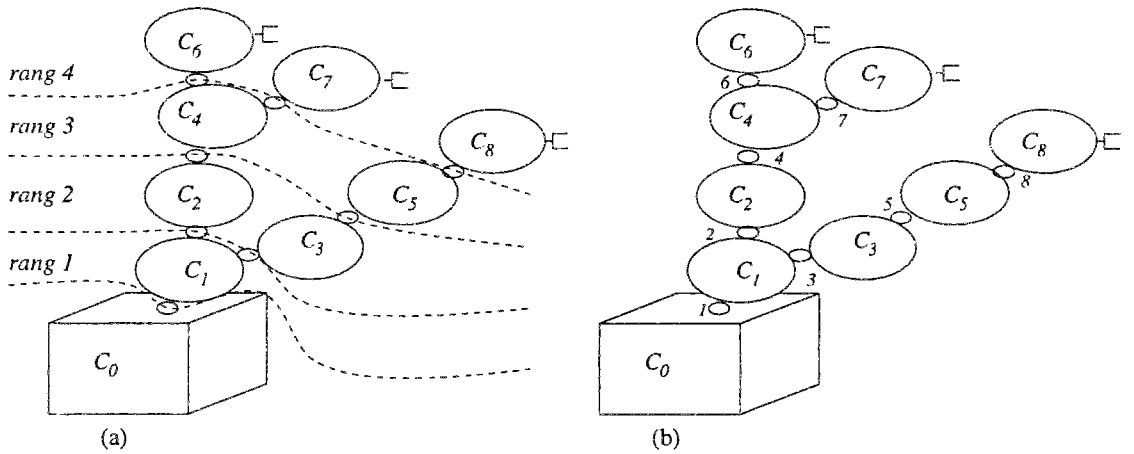
Soit un mécanisme à structure topologique d'arbre et à N degrés de liberté. Si i^+ et i^- désignent ses deux applications indicatrices, alors:

$$\begin{cases} \forall k \in I_N : & i^+(k) = \{j \in I_N^* / u_{kj} = 1\} \\ \forall k \in I_N^* : & i^-(k) = j \Leftrightarrow u_{jk} = 1 \end{cases} \qquad \text{(Eq.3.10)}$$

[6]Dans l'étude topologique du mécanisme, on ne s'occupe jamais de la nature des articulations mécaniques. Dans le cas où l'on dispose d'un mécanisme dont toutes les articulations sont simples (prismatiques ou rotoïdes), le nombre N désignera le nombre de degrés de liberté de la structure articulée.

3.5.3 Exemple de structure arborescente

Considérons le mécanisme donné par la figure Fig. 3.11. Il contient 8 éléments en plus du socle,



Dans l'exemple présent, le robot dispose de trois organes terminaux. La numérotation de la structure arborescente va pouvoir être effectuée en deux étapes schématisées par les figures (a) et (b):

(a) Déterminations de l'ordre des éléments de la structure articulée et leur indiçage suivant les règles de la définition 3.7.

(b) Indiçage des articulations de la structure arborescente suivant les règles de la définition 3.7.

Fig. 3.11: Ordre récursivement reconnaissable sur un arbre topologique

8 degrés de liberté et 3 organes terminaux. La matrice des successeurs immédiats des éléments de cette structure est:

$$\Upsilon = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Ainsi, en notant l'ensemble vide par $\emptyset$, les indicatrices de la structure arborescente de la figure Fig. 3.11 sont données par le tableau 3.2.

Tableau 3.2 : *Indicatrices de la structure arborescente de la figure Fig. 3.11*

Corps d'indice k	0	1	2	3	4	5	6	7	8
Rang du corps C_k	0	1	2	2	3	3	4	4	4
$i^+(k)$	$\{1\}$	$\{2,3\}$	$\{4\}$	$\{5\}$	$\{6,7\}$	$\{8\}$	$\emptyset$	$\emptyset$	$\emptyset$
$i^-(k)$	indéfini	0	1	1	2	3	4	4	5

3.6 Variétés d'étude des systèmes articulés

L'intérêt de savoir situer un solide dans l'espace s'est manifesté, tout d'abord, à propos de la mécanique céleste, puis ces méthodes ont été généralisées à l'étude des mécanismes et plus tardivement à la robotique, avec la naissance de cette discipline, dans les années cinquante. L'étude des robots et mécanismes fait appel à deux sortes de variétés.

3.6.1 Espace opérationnel

On a l'habitude de définir l'espace opérationnel d'un robot ou d'un mécanisme comme étant la variété dans laquelle sont représentées les configurations de son organe terminal. Plusieurs solutions peuvent être proposées pour la représentation d'une telle variété. Celle que nous préconisons consiste à décrire une configuration grâce à des déplacements indépendants relatifs aux axes d'une famille fondamentale -de référence- dans un système de coordonnées donné. Si m désigne le nombre des paramètres nécessaires à la description de l'espace des configurations, la valeur m constitue alors le nombre de degrés de liberté maximum que peut avoir l'organe terminal. Etant donné que le nombre de degrés de liberté d'un solide isolé est 6, alors $0 \leq m \leq 6$.

3.6.2 Espace articulaire

Par opposition à l'espace opérationnel qui définit le mouvement absolu de l'organe terminal, l'espace articulaire permet de décrire les mouvements relatifs des différents éléments, d'un robot ou d'un système articulé, par rapport à leurs prédécesseurs immédiats.

Définition 3.8

- *On appelle configuration articulaire d'un robot manipulateur ou d'un système articulé, la collection* $(s_1, s_2, \cdots, s_N)$ *des configurations de ses différents éléments (où* s_i *est la configuration du corps* C_i*).*
- *Ces configurations peuvent être décrites par coordonnées articulaires (qui définissent le d.d.l. de chaque élément par rapport à son prédécesseur dans la chaîne articulée). L'espace engendré par ces coordonnées est appelé espace articulaire.*
- *Le nombre de variables articulaires -réelles- indépendantes*[7] *entre elles est dit degrés de liberté du robot.*
- *On dit qu'un robot ou un système articulé est redondant, si le nombre de degrés de liberté de l'organe terminal est inférieur au nombre des articulations motorisées.*

[7] En général, dans les structures ouvertes simples (ou arborescentes), les variables articulaires sont indépendantes, tandis que dans les structures qui comportent une ou plusieurs boucles fermées impérativement, il existe un nombre de relations qui relient ces variables entre elles.

Pour une chaîne ouverte simple, on peut se retrouver dans le cas d'une structure redondante, par exemple si:

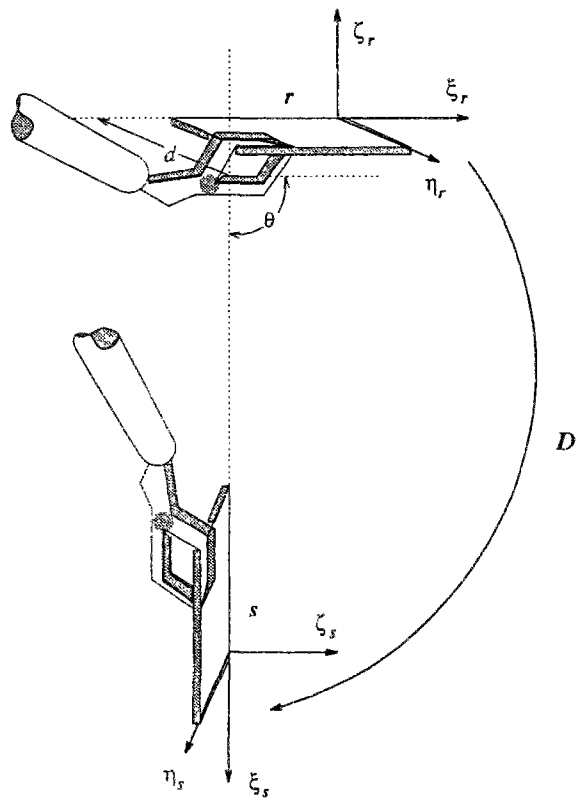
- Deux articulations prismatiques dont les générateurs infinitésimaux des mouvements articulaires sont $\boldsymbol{R}$–proportionnels (i.e. articulations prismatiques parallèles).
- Plus de trois articulations prismatiques (puisque $\mathfrak{t}$ est un $\boldsymbol{R}$–espace vectoriel de dimension 3).
- Deux articulations rotoïdes dont les générateurs infinitésimaux s'annulent simultanément en plus d'un point de l'espace $\mathcal{E}$ (i.e. axes confondus).
- Plus de trois articulations rotoïdes dont les générateurs infinitésimaux s'annulent simultanément en un même point de l'espace (i.e. axes concourants).
- Plus de trois articulations rotoïdes dont les vecteurs sommes des générateurs infinitésimaux sont équipollent (i.e. axes parallèles).
- Plus de six articulations motorisées (puisque $\mathfrak{d}$ est un $\boldsymbol{R}$–espace vectoriel de dimension 6).

Dans les problèmes de planification des trajectoires en présence d'obstacles, la redondance permet d'augmenter l'accessibilité de l'organe terminal. Dans de tels cas les robots doivent satisfaire différents types de critères d'optimisation sur des contraintes d'anti-collision entre autres [P. Tournassoud [TOUR-88], E. Dombre & W. Khalil [DOMB-88], J. Angeles, A. Alivizatos & P.J. Zsombor-Muray [ANGE-90]]. Le choix du nombre de degrés de liberté minimal dont doit disposer le robot peut être déterminé en fonction de la tâche prévue. En fait, ce nombre de degrés de liberté doit être supérieur à celui de la tâche pour que le mécanisme puisse placer son organe terminal dans la configuration désirée. Par ailleurs, les caractéristiques géométriques de l'objet manipulé peuvent aider à la détermination du nombre de degrés de liberté. Un robot non redondant doit disposer de six degrés de liberté pour situer un corps quelconque dans une configuration quelconque. Si le corps présente une symétrie de révolution, alors, cinq degrés de liberté suffisent, puisqu'il devient inutile de spécifier la rotation autour de l'axe de révolution du solide. Seulement trois degrés de liberté en translation sont nécessaires pour le positionnement d'un corps qui présente une symétrie sphérique.

3.7 Cosinus directeurs duaux

Généralement nous pouvons, non seulement, construire une famille $\mathfrak{f}_r$ rigidement solidaire à une configuration r donnée de l'organe terminal; mais aussi, décrire les configurations de celui-ci dans l'espace relativement à une configuration de référence. En effet, la description d'une configuration s par rapport à une configuration de référence r peut être traduite en fonction du déplacement $\boldsymbol{D}$ tel que:

$$s = \boldsymbol{D} \bullet r$$



Cet figure schématise le changement de familles fondamentales associé au changement de configurations de l'organe terminal d'un manipulateur. A titre d'exemple, l'angle dual $\alpha = arccos\{\xi_s/\xi_r\} = \theta + \varepsilon d$ correspond au cosinus directeur $\{\xi_s/\xi_r\}$, où θ est l'angle formé par les directions des axes centraux Λ_{ξ_r} et Λ_{ξ_s} et d est la distance de de l'axe Λ_{ξ_r} à l'axe Λ_{ξ_s}.

Fig. 3.12: Positionnement de l'organe terminal

Le déplacement $\boldsymbol{D}$, ainsi obtenu, décrit la transformation géométrique qui change la configuration $\boldsymbol{r}$ en $\boldsymbol{s}$. Si la représentation adjointe du déplacement $\boldsymbol{D}$ est parfaitement connue, alors la famille $\mathfrak{f}_s$ peut être calculée:

$$\mathfrak{f}_s = \boldsymbol{D} \diamond \mathfrak{f}_r$$

c'est à dire:

$$\xi_s = \boldsymbol{D}_* \xi_r,\ \eta_s = \boldsymbol{D}_* \eta_r \text{ et } \zeta_s = \boldsymbol{D}_* \zeta_r$$

Maintenant, si les coordonnées des éléments de la famille $\mathfrak{f}_s$, exprimées dans la famille $\mathfrak{f}_r$, sont parfaitement connues; alors, l'expression de la matrice duale $[\boldsymbol{D}_*]$ (d'ordre 3×3) associée à l'opérateur $\boldsymbol{D}_*$, relativement à la base $\mathfrak{f}_r$, est définie par:

$$[\boldsymbol{D}_*] = \begin{pmatrix} \{\xi_s/\xi_r\} & \{\eta_s/\xi_r\} & \{\zeta_s/\xi_r\} \\ \{\xi_s/\eta_r\} & \{\eta_s/\eta_r\} & \{\zeta_s/\eta_r\} \\ \{\xi_s/\zeta_r\} & \{\eta_s/\zeta_r\} & \{\zeta_s/\zeta_r\} \end{pmatrix} \qquad \text{(Eq.3.11)}$$

En résumé, comme les éléments des familles fondamentales $\mathfrak{f}_r$ et $\mathfrak{f}_s$ sont unitaires pour le produit scalaire dual, le produit scalaire des éléments de la famille $\mathfrak{f}_s$ par ceux de la famille $\mathfrak{f}_r$ définit les cosinus (duaux) des angles duaux formés par les deux axes des glisseurs de chaque produit [Fig. 3.12]. Ainsi, nous dirons que les composantes de la matrice $[\boldsymbol{D}_*]$ sont les cosinus directeurs duaux relatifs aux familles $\mathfrak{f}_r$ et $\mathfrak{f}_s$ pour le produit scalaire dual sur $\mathfrak{D}$.

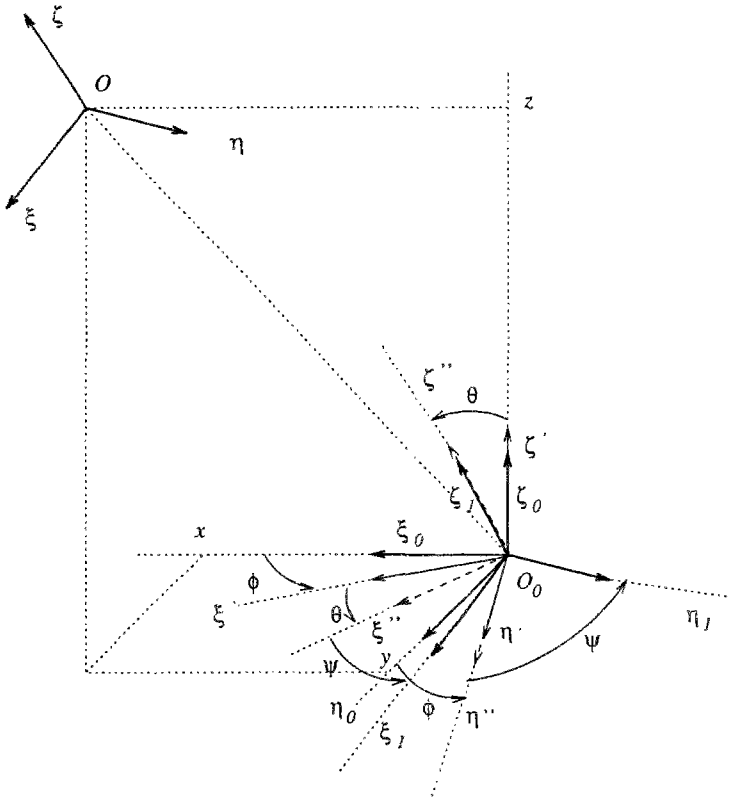
Enfin, lorsque $s(t)$ décrit la trajectoire associée à un mouvement donnée par l'application $t \mapsto \boldsymbol{D}(t)$, les assignations $\boldsymbol{D}(t)$ décrivent le mouvement de la famille mobile $\mathfrak{f}_s$ par rapport à la famille de référence $\mathfrak{f}_r$. En se reportant à la relation (Eq.3.11), on voit bien que la $j^{ème}$ colonne (où $j = 1..3$) de la matrice $[\boldsymbol{D}_*(t)]$ décrit les variations en fonction du temps du $j^{ème}$ élément de la famille $\mathfrak{f}_s$ relativement à la base $\mathfrak{f}_r$ du Δ–module libre $\mathfrak{D}$.

3.8 Systèmes de coordonnées

Dans les problèmes de mécanique classique, pour situer un repère $\mathcal{R} = (O; \vec{i}, \vec{j}, \vec{k})$ par rapport à repère de référence $\mathcal{R}_0 = (O_0; \vec{i}_0, \vec{j}_0, \vec{k}_0)$, on a l'habitude de donner [Fig. 3.13]:

- **La position**: Elle est définie par la position du point O dans le repère $\mathcal{R}_0$ décrite par trois coordonnées -cartésiennes, cylindriques, sphériques ou autres- (on choisit le système de coordonnées le mieux adapté au problème géométrique en fonction de la structure de base du manipulateur),
- **L'orientation**: L'orientation du repère $\mathcal{R}$ par rapport à $\mathcal{R}_0$ revient à représenter une rotation $\boldsymbol{R}$ d'angle ψ autour d'un axe de vecteur unitaire $\vec{u}$ permettant, à une translation près, de faire coïncider le repère $\mathcal{R}_0$ avec $\mathcal{R}$. Si $\vec{u} = (u_x, u_y, u_z)^T$ sont les coordonnées du vecteur $\vec{u}$ relativement à $\mathcal{R}_0$, là aussi plusieurs choix sont possibles, pour la description d'une telle rotation. A titre d'exemple:
  - **Les paramètres d'Euler ou d'Olinde–Rodrigues**: un jeu de quatre paramètres p, q, r et s (i.e. coordonnées du quaternion représentant la rotation $\boldsymbol{R}$) définis par:

$$p = u_x \sin\frac{\psi}{2},\ q = u_y \sin\frac{\psi}{2},\ r = u_z \sin\frac{\psi}{2},\ s = \cos\frac{\psi}{2} \qquad \text{(Eq.3.12)}$$



Euler avait montrer qu'il suffisait, à une translation près, de trois rotations pour amener un trièdre de référence à entrer en coïncidence avec un second trièdre d'orientation quelconque lié à un solide dans l'espace:

- Orientation: comme nous avons montrer que la notion de repère affine coïncidait avec celle de famille fondamentale. Exprimons l'orientation grâce aux angles d'Euler classiques dans le langage des familles fondamentales:
  - on transforme la famille $\mathfrak{f}_0$ en $\mathfrak{f}'$ par rotation d'un angle ϕ autour de l'axe Λ_{ζ_0},
  - ensuite, on transforme la famille $\mathfrak{f}'$ en $\mathfrak{f}''$ par rotation d'un angle θ autour de l'axe $\Lambda_{\eta'}$,
  - enfin, on transforme la famille $\mathfrak{f}''$ en $\mathfrak{f}_1$ par rotation d'un angle ψ autour de l'axe $\Lambda_{\zeta''}$.
- Position: on calcul les coordonnées cartésiennes (x, y, z) du point O (origine de la famille $\mathfrak{f}$ dans la famille $\mathfrak{f}_0$. La translation correspondant au positionnement étant la translation de vecteur $\overrightarrow{O_0 O}$ de la famille $\mathfrak{f}_1$.

Fig. 3.13: Angles d'Euler classiques

ces quatre paramètres sont liés par une relation de normalité:

$$p^2 + q^2 + r^2 + s^2 = 1 \qquad \text{(Eq.3.13)}$$

l'intérêt de cette paramétrisation est qu'elle ne présente pas de singularités. Par contre elle comporte l'inconvénient de représenter 3 d.d.l. par quatre paramètres liés entre eux par une relation de dépendance exprimée par (Eq.3.13).

- **Les paramètres de Rodrigues**: donnés par l'ensemble des trois paramètres:

$$e_1 = u_x\, tg\,\frac{\psi}{2},\ e_2 = u_y\, tg\,\frac{\psi}{2},\ e_3 = u_z\, tg\,\frac{\psi}{2} \qquad \text{(Eq.3.14)}$$

L'intérêt, de cette paramétrisation, est qu'elle offre une représentation de l'orientation avec trois paramètres indépendants. Par contre elle comporte l'inconvénient présenter des singularités pour $\psi = \pm\pi$.

D'autres représentations sont possibles en utilisant les angles d'Euler [Fig. 3.13], de Briant ou autres.

Dans la théorie générale des groupes et algèbres de Lie, on utilise plutôt la notion de coordonnées de première et seconde espèce. Dans cet ordre d'idée et dans l'état actuel de nos développements, il est naturel de se demander:

i) Est-il possible de décrire un déplacement $\boldsymbol{D}$ quelconque par la composition de trois déplacements cylindriques?

ii) Si une telle description est possible, peut-on généraliser les descriptions classiques des orientations par les angles d'Euler ou de Briant (cartes de $SO(3)$) à des déplacements quelconques en utilisant cette fois des cartes -sur les nombres duaux- du groupe de Lie $\mathbb{D}$?
(les descriptions classiques par les angles de Briant et d'Euler pourront être retrouvées en considérant tout simplement des rotations au lieu de déplacements cylindriques dans cette éventuelle description).

iii) Quel intérêt pratique présenterait le recours à de telles généralisations de conventions?

La réponse à la question *i*) est *oui*. C'est en fait la décomposition associée aux coordonnées de seconde espèce dans la base $(\xi, \Omega\xi, \eta, \Omega\eta, \zeta, \Omega\zeta)$ de $\mathfrak{d}$ liée à une famille fondamentale $\mathfrak{f} = (\xi, \eta, \zeta)$. La réponse à la question *ii*) fera l'objet des paragraphes 3.8.1 et 3.8.2. La méthode de généralisation des conventions d'Euler et de Briant, que nous y proposons consiste à se donner deux familles fondamentales $\mathfrak{f}_1$ et $\mathfrak{f}_2$ et procéder ensuite suivant les étapes:

(a) partir de la famille fondamentale $\mathfrak{f}_1$,

(b) procéder par trois déplacements cylindriques indépendants avec des conventions, de transformation des familles fondamentales successives, semblables à celles proposées dans la littérature pour les rotations par des angles de Briant ou d'Euler, à la différence près que chaque fois que l'on fera "tourner" une famille par une rotation autour de l'axe de l'un de ses éléments, on suivra cette rotation par une translation le long du même axe.

(c) arriver à la famille fondamentale $\mathfrak{f}_2$.

Pour chacune des méthodes proposées, nous allons commencer d'abord par décrire l'algorithme des transformations correspondantes, puis nous allons identifier les paramètres intervenant dans ces descriptions et donner les conditions d'application de telles méthodes.

Quant à l'intérêt de nos méthodes, c'est qu'elles vont permettre de faire un calcul symbolique très simple à partir de la théorie des groupes en utilisant l'outil formel des nombres duaux qui permet de simplifier la syntaxe des modèles. Le calcul sur les paramètres réels du modèle pourra donc être obtenu de façon automatique en séparant les parties réelles et duales des expressions duales.

3.8.1 Angles de Briant duaux: Roulis, Tangage et Lacet duaux

3.8.1.a Algorithme

(a) Appelons Roulis dual $\phi = \alpha + \varepsilon\, a$ et appliquons le déplacement cylindrique $\boldsymbol{D}_\phi$ d'angle dual ϕ suivant l'axe central de ξ_1. Il transforme alors la famille $\mathfrak{f}_1$ en une deuxième famille $\mathfrak{f}'$. $\boldsymbol{D}_\phi$ s'exprime alors par:

$$\boldsymbol{D}_\phi = \quad exp\,(\phi\,\xi_1) \tag{Eq.3.15}$$

La matrice duale associée à la représentation adjointe de $\boldsymbol{D}_\phi$ relativement à la famille $\mathfrak{f}_1$, est donnée par:

$$[\boldsymbol{D}_{\phi *}] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\phi & -sin\,\phi \\ 0 & sin\,\phi & cos\,\phi \end{pmatrix} \tag{Eq.3.16}$$

(b) Appelons Tangage dual $\theta = \beta + \varepsilon\, b$ et, de la même façon, appliquons à la famille $\mathfrak{f}'$ le déplacement $\boldsymbol{D}_\theta$ d'angle dual θ suivant l'axe de η_1. Une troisième famille $\mathfrak{f}''$ est obtenue au terme de cette transformation. Donc:

$$\boldsymbol{D}_\theta = \quad exp\,(\theta\,\eta') \tag{Eq.3.17}$$

la matrice associée à sa représentation adjointe s'exprime, alors, par:

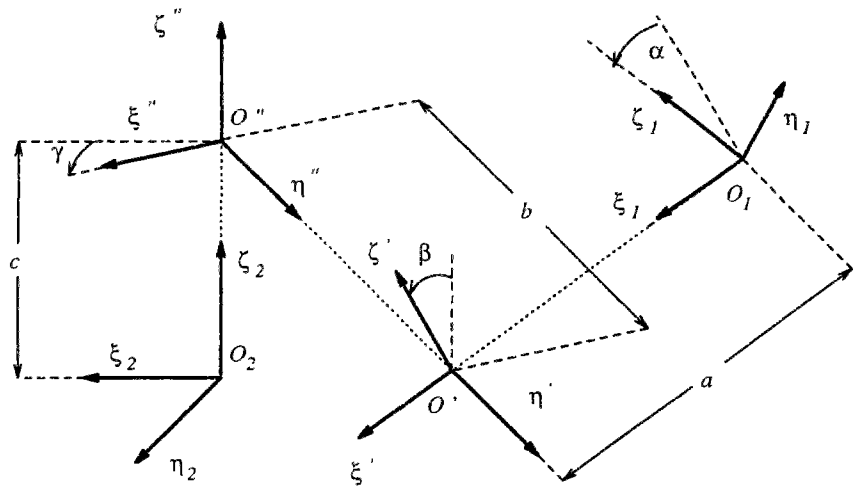
$$[\boldsymbol{D}_{\theta *}] = \begin{pmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{pmatrix} \tag{Eq.3.18}$$

(c) Appelons maintenant Lacet dual $\psi = \gamma + \varepsilon\, c$. A la nouvelle famille $\mathfrak{f}''$ nous allons appliquer le déplacement $\boldsymbol{D}_\psi$ d'angle dual ψ suivant l'axe du glisseur ζ'':

$$\boldsymbol{D}_\psi = \quad exp\,(\psi\,\zeta'') \tag{Eq.3.19}$$

et la matrice associée à sa représentation adjointe s'exprime par:

$$[\boldsymbol{D}_{\psi *}] = \begin{pmatrix} cos\,\psi & -sin\,\psi & 0 \\ sin\,\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{Eq.3.20}$$



Schématisation du paramétrage, d'un déplacement quelconque, correspondant aux angles roulis, tangage et lacet duaux. Chaque rotation duale, autour d'un élément de la famille fondamentale en présence, se traduit par une rotation réelle autour de son axe central et une translation de longueur finie le long du même axe.

Fig. 3.14: Angles de Briant duaux

La composition des trois déplacements décrits par cet algorithme permet d'écrire:

$$\boldsymbol{D} = \quad exp\,(\psi\ \zeta'')\ exp\,(\theta\ \eta')\ exp\,(\phi\ \xi_1)$$

que nous noterons plus simplement:

$$\boldsymbol{D} = \quad \boldsymbol{D}_\psi\ \boldsymbol{D}_\theta\ \boldsymbol{D}_\phi \qquad \text{(Eq.3.21)}$$

La matrice duale associée à la représentation adjointe d'un tel déplacement $\boldsymbol{D}$ est donnée par la relation:

$$[D_*] = \begin{pmatrix} cos\,\psi & -sin\,\psi & 0 \\ sin\,\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\phi & -sin\,\phi \\ 0 & sin\,\phi & cos\,\phi \end{pmatrix} \qquad \text{(Eq.3.22)}$$

c'est à dire encore:

$$[D_*] = \begin{pmatrix} cos\,\psi\,cos\,\theta & cos\,\psi\,sin\,\theta\,sin\,\phi - sin\,\psi\,cos\,\phi & cos\,\psi\,sin\,\theta\,cos\,\phi + sin\,\psi\,sin\,\phi \\ sin\,\psi\,cos\,\theta & sin\,\psi\,sin\,\theta\,sin\,\phi + cos\,\psi\,cos\,\phi & sin\,\psi\,sin\,\theta\,cos\,\phi - cos\,\psi\,sin\,\phi \\ -sin\,\theta & cos\,\theta\,sin\,\phi & cos\,\theta\,cos\,\phi \end{pmatrix} \qquad \text{(Eq.3.23)}$$

où les symboles *cos* et *sin* désignent les prolongements canoniques duaux des fonctions cosinus et sinus au sens trigonométrique classique.

Les parties réelle et duale de cette matrice sont données par les matrices réelles[8]:

$$Re[D_*] = \begin{pmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \cos\gamma\sin\beta\cos\alpha + \sin\gamma\sin\alpha \\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha + \cos\gamma\cos\alpha & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{pmatrix}$$

et

$$Du[D_*] = \begin{pmatrix} 1^e \text{ colonne} & \begin{cases} -\cos\gamma\, b\sin\beta - c\sin\gamma\cos\beta \\ -\sin\gamma\, b\sin\beta + c\cos\gamma\cos\beta \\ -b\cos\beta \end{cases} \\ 2^e \text{ colonne} & \begin{cases} \cos\gamma\sin\beta\, a\cos\alpha + \cos\gamma\, b\cos\beta\sin\alpha - c\sin\gamma\sin\beta\sin\alpha + \sin\gamma\, a\sin\alpha - c\cos\gamma\cos\alpha \\ \sin\gamma\sin\beta\, a\cos\alpha + \sin\gamma\, b\cos\beta\sin\alpha + c\cos\gamma\sin\beta\sin\alpha - \cos\gamma\, a\sin\alpha - c\sin\gamma\cos\alpha \\ \cos\beta\, a\cos\alpha - b\sin\beta\sin\alpha \end{cases} \\ 3^e \text{ colonne} & \begin{cases} -\cos\gamma\sin\beta\, a\sin\alpha + \cos\gamma\, b\cos\beta\cos\alpha - c\sin\gamma\sin\beta\cos\alpha + \sin\gamma\, a\cos\alpha + c\cos\gamma\sin\alpha \\ -\sin\gamma\sin\beta\, a\sin\alpha + \sin\gamma\, b\cos\beta\cos\alpha + c\cos\gamma\sin\beta\cos\alpha - \cos\gamma\, a\cos\alpha + c\sin\gamma\sin\alpha \\ -\cos\beta\, a\sin\alpha - b\sin\beta\cos\alpha \end{cases} \end{pmatrix}$$

3.8.1.b Identification des paramètres et singularités

Jusqu'à présent nous n'avons fait que décrire la généralisation des conventions des angles de Briant. Pour pouvoir justifier ce paramétrage, il est nécessaire de résoudre le problème inverse. Le problème d'inversion consiste à déterminer les angles duaux (des transformations cylindriques) qui correspondent à cette décomposition. Ainsi, si:

$$[\boldsymbol{D}_*] = \begin{pmatrix} \delta_{1\,1} & \delta_{1\,2} & \delta_{1\,3} \\ \delta_{2\,1} & \delta_{2\,2} & \delta_{2\,3} \\ \delta_{3\,1} & \delta_{3\,2} & \delta_{3\,3} \end{pmatrix} \qquad \text{(Eq.3.24)}$$

Cette matrice peut, éventuellement, être la matrice des cosinus directeurs duaux correspondant au passage de la famille $\mathfrak{f}_1$ à la famille $\mathfrak{f}_2$ de type (Eq.3.11). La solution dépend en fait de la résolution dans Δ d'un ensemble de 9 équations transcendantes qui expriment l'égalité entre les composantes des matrices (Eq.3.23) et (Eq.3.24). Ainsi, la solution sous sa forme duale est donnée par:

$$\begin{cases} \theta = arctg\left(\dfrac{-\delta_{3\,1}}{\sqrt{\delta_{1\,1}^2 + \delta_{2\,1}^2}}\right) \\ \psi = arctg\left(\dfrac{\delta_{2\,1}/cos\,\theta}{\delta_{1\,1}/cos\,\theta}\right) \\ \phi = arctg\left(\dfrac{\delta_{3\,2}/cos\,\theta}{\delta_{3\,3}/cos\,\theta}\right) \end{cases} \qquad \text{(Eq.3.25)}$$

L'existence du prolongement canonique de la fonction trigonométrique inverse *arctg* est assurée par le théorème 1.7 que nous avons établi au chapitre 1 [page 21]. L'extraction des parties réelles et duales de θ, ψ et ϕ permet de retrouver les six paramètres réels qui décrivent une telle

[8] Ce calcul à été effectué grâce au module `dualstruc.map` que nous avons implémenté dans le langage de Maple.

décomposition. Ainsi, si nous posons $r_{ij} = Re\,\delta_{ij}$ et $d_{ij} = Du\,\delta_{ij}$ pour $i, j \in [1..3]$, alors:

$$\left\{\begin{array}{l} \gamma = arctg\left(\dfrac{r_{21}}{r_{11}}\right) \\ \\ \beta = arctg\left(\dfrac{-r_{31}}{\sqrt{r_{11}^2 + r_{21}^2}}\right) \\ \\ \alpha = arctg\left(\dfrac{r_{32}}{r_{33}}\right) \\ \\ c = \dfrac{-r_{11}\,d_{21} + d_{11}\,r_{21}^3}{r_{11}^2 + r_{21}^2} \\ \\ b = -\dfrac{-r_{31}\,d_{11}\,r_{11} - r_{31}\,d_{21}\,r_{21} + d_{31}\,r_{11}^4 + 2\,d_{31}\,r_{11}^2 r_{21}^2 + d_{31}\,r_{21}^4}{\sqrt{r_{11}^2 + r_{21}^2}\,(r_{31}^2 + r_{11}^2 + r_{21}^2)} \\ \\ a = \dfrac{-r_{32}\,d_{33} + d_{32}\,r_{33}^3}{r_{32}^2 + r_{33}^2} \end{array}\right. \qquad \text{(Eq.3.26)}$$

D'après les relations (Eq.3.25) qui expriment les angles duaux ϕ et ψ, la méthode que nous venons de décrire présente certaines singularités lorsque la quantité duale $cos\,\theta$ n'est pas inversible (division par un nombre dual), c'est à dire, lorsque $Re(cos\,\theta) = 0$. En effet, la solution dégénère lorsque:

$$Re(\theta) = \quad \beta = \quad \pm\frac{\pi}{2} \qquad \text{(Eq.3.27)}$$

auquel cas les deux transformations intermédiaires se retrouvent colinéaires et le déplacement total se résume à une rotation duale autour de l'axe ζ. Ce déplacement s'exprime alors par:

$$\boldsymbol{D} = \quad exp\left((\psi \pm \phi)\,\zeta\right) \qquad \text{(Eq.3.28)}$$

Les configurations singulières sont alors données par les valeurs du tangage dual θ dans l'ensemble $\pm\frac{\pi}{2} \oplus \varepsilon\,\boldsymbol{R}$ (i.e. $-\frac{\pi}{2} \oplus \varepsilon\,\boldsymbol{R} \cup \frac{\pi}{2} \oplus \varepsilon\,\boldsymbol{R}$).

3.8.2 Angles d'Euler duaux: Précession - Nutation et Rotation propre duales

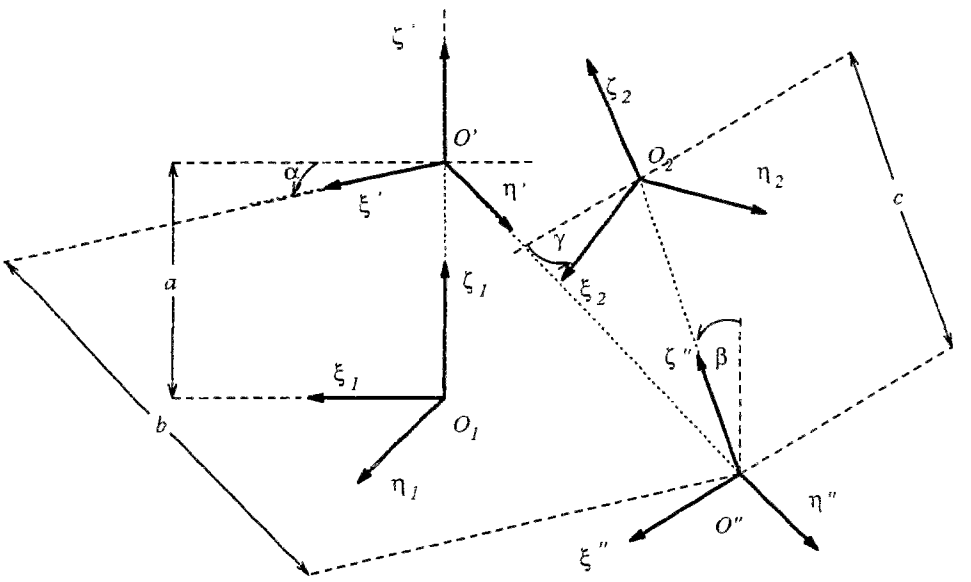
3.8.2.a Algorithme

(a) Appelons précession duale $\phi = \alpha + \varepsilon\,a$. Dans un premier temps, appliquons à la famille $\mathfrak{f}_1$ le déplacement $\boldsymbol{D}_\phi$ d'angle dual ϕ suivant l'axe de ζ_1 et notons $\boldsymbol{D}_\phi$ ce déplacement, il s'écrit:

$$\boldsymbol{D}_\phi = \quad exp\left(\phi\,\zeta_1\right) \qquad \text{(Eq.3.29)}$$

la matrice associée à sa représentation adjointe s'exprime par:

$$[\boldsymbol{D}_{\phi_*}] = \begin{pmatrix} cos\,\phi & -sin\,\phi & 0 \\ sin\,\phi & cos\,\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{(Eq.3.30)}$$



Schématisation du paramétrage, d'un déplacement quelconque, correspondant aux angles d'Euler duaux. Comme dans le cas précédent, chaque rotation duale, autour d'un élément de la famille fondamentale en présence, se traduit par une rotation réelle autour de son axe central et une translation de longueur finie le long du même axe.

Fig. 3.15: Paramétrage par des angles d'Euler duaux

(b) Appelons nutation duale $\theta = \beta + \varepsilon\, b$ et appliquons à la famille $\mathfrak{f}'$ le déplacement $\boldsymbol{D}_\theta$ d'angle dual θ suivant l'axe de η'. Alors:

$$\boldsymbol{D}_\theta = \quad exp\,(\theta\,\eta') \tag{Eq.3.31}$$

la matrice associée à sa représentation adjointe est donc:

$$[\boldsymbol{D}_{\theta *}] = \begin{pmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{pmatrix} \tag{Eq.3.32}$$

(c) Enfin, appelons rotation propre duale $\psi = \gamma + \varepsilon\, c$ et appliquons à la nouvelle famille $\mathfrak{f}''$ le déplacement $\boldsymbol{D}_\psi$ d'angle dual ψ autour de l'axe de ζ'' pour la faire coïncider avec la famille $\mathfrak{f}_2$:

$$\boldsymbol{D}_\psi = exp(\psi\, \zeta_1) \qquad \text{(Eq.3.33)}$$

la matrice associée à sa représentation adjointe est:

$$[\boldsymbol{D}_{\psi_*}] = \begin{pmatrix} cos\,\psi & -sin\,\psi & 0 \\ sin\,\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{(Eq.3.34)}$$

La composition des trois déplacements décrits ci-avant permet d'écrire:

$$\boldsymbol{D} = exp(\psi\, \zeta'')\, exp(\theta\, \eta')\, exp(\phi\, \zeta_1) \qquad \text{(Eq.3.35)}$$

ou tout simplement:

$$\boldsymbol{D} = \boldsymbol{D}_\psi\, \boldsymbol{D}_\theta\, \boldsymbol{D}_\phi \qquad \text{(Eq.3.36)}$$

si aucune confusion n'est à craindre. La matrice duale associée à la représentation adjointe d'un tel déplacement est donnée par la relation:

$$[D_*] = \begin{pmatrix} cos\,\psi & -sin\,\psi & 0 \\ sin\,\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{pmatrix} \begin{pmatrix} cos\,\phi & -sin\,\phi & 0 \\ sin\,\phi & cos\,\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{(Eq.3.37)}$$

soit encore:

$$[D_*] = \begin{pmatrix} cos\,\psi\, cos\,\theta\, cos\,\phi - sin\,\psi\, sin\,\phi & -cos\,\psi\, cos\,\theta\, sin\,\phi - sin\,\psi\, cos\,\phi & cos\,\psi\, sin\,\theta \\ sin\,\psi\, cos\,\theta\, cos\,\phi + cos\,\psi\, sin\,\phi & -sin\,\psi\, cos\,\theta\, sin\,\phi + cos\,\psi\, cos\,\phi & sin\,\psi\, sin\,\theta \\ -sin\,\theta\, cos\,\phi & sin\,\theta\, sin\,\phi & cos\,\theta \end{pmatrix} \qquad \text{(Eq.3.38)}$$

Les parties réelle et duale d'une telle matrice duale sont données respectivement par:

$$Re\,[D_*] = \begin{pmatrix} \cos\gamma\, \cos\beta\, \cos\alpha - \sin\gamma\, \sin\alpha & -\cos\gamma\, \cos\beta\, \sin\alpha - \sin\gamma\, \cos\alpha & \cos\gamma\, \sin\beta \\ \sin\gamma\, \cos\beta\, \cos\alpha + \cos\gamma\, \sin\alpha & -\sin\gamma\, \cos\beta\, \sin\alpha + \cos\gamma\, \cos\alpha & \sin\gamma\, \sin\beta \\ -\sin(\beta)\, \cos\alpha & \sin\beta\, \sin\alpha & \cos\beta \end{pmatrix}$$

et

$$Du\,[D_*] = \begin{pmatrix} 1^e \text{ colonne} & \begin{cases} -\cos\gamma\, \cos\beta\, a\, \sin\alpha - \cos\gamma\, b\, \sin\beta\, \cos\alpha - c\, \sin\gamma\, \cos\beta\, \cos\alpha - \sin\gamma\, a\, \cos\alpha - c\, \cos\gamma\, \sin\alpha \\ -\cos\gamma\, \cos\beta\, a\, \cos\alpha + \cos\gamma\, b\, \sin\beta\, \sin\alpha + c\, \sin\gamma\, \cos\beta\, \sin\alpha + \sin\gamma\, a\, \sin\alpha - c\, \cos\gamma\, \cos\alpha \\ \cos\gamma\, b\, \cos\beta - c\, \sin\gamma\, \sin\beta \end{cases} \\ 2^e \text{ colonne} & \begin{cases} -\sin\gamma\, \cos\beta\, a\, \sin\alpha - \sin\gamma\, b\, \sin\beta\, \cos\alpha + c\, \cos\gamma\, \cos\beta\, \cos\alpha + \cos\gamma\, a\, \cos\alpha - c\, \sin\gamma\, \sin\alpha \\ -\sin\gamma\, \cos\beta\, a\, \cos\alpha + \sin\gamma\, b\, \sin\beta\, \sin\alpha - c\, \cos\gamma\, \cos\beta\, \sin\alpha - \cos\gamma\, a\, \sin\alpha - c\, \sin\gamma\, \cos\alpha \\ \sin\beta\, a\, \cos\alpha + b\, \cos\beta\, \sin\alpha \end{cases} \\ 3^e \text{ colonne} & \begin{cases} \sin\beta\, a\, \sin\alpha - b\, \cos\beta\, \cos\alpha \\ \sin\gamma\, b\, \cos\beta + c\, \cos\gamma\, \sin\beta \\ -b\, \sin\beta \end{cases} \end{pmatrix}$$

3.8.2.b Identification des paramètres et singularités

Comme pour la méthode précédente, l'inversion du positionnement décrit ci-avant dépend de la résolution d'un ensemble de 9 équations transcendantes qui expriment l'égalité entre les matrices (Eq.3.38) et (Eq.3.24). La solution sous sa forme duale est donnée par:

$$\begin{cases} \theta = arctg\left(\dfrac{\sqrt{\delta_{31}^2 + \delta_{32}^2}}{\delta_{33}}\right) \\ \psi = arctg\left(\dfrac{\delta_{23}/sin\,\theta}{\delta_{13}/sin\,\theta}\right) \\ \phi = arctg\left(\dfrac{\delta_{32}/sin\,\theta}{-\delta_{31}/sin\,\theta}\right) \end{cases} \qquad \text{(Eq.3.39)}$$

L'extraction des parties réelles et duales permet de retrouver alors les six paramètres réels qui décrivent une telle décomposition:

$$\begin{cases} \gamma = arctg(\dfrac{r_{23}}{r_{13}}) \\ \beta = arctg(\dfrac{\sqrt{{r_{31}}^2 + {r_{32}}^2}}{r_{33}}) \\ \alpha = -arctg(\dfrac{r_{32}}{r_{31}}) \\ c = -\dfrac{r_{23}\,d_{13} - d_{23}\,r_{13}}{{r_{13}}^2 + {r_{23}}^2} \\ b = -\dfrac{{r_{31}}^2 d_{33} + {r_{32}}^2 d_{33} - r_{31}\,d_{31}\,r_{33} - r_{32}\,d_{32}\,r_{33}}{\sqrt{{r_{31}}^2 + {r_{32}}^2}\,({r_{33}}^2 + {r_{31}}^2 + {r_{32}}^2)} \\ a = \dfrac{r_{32}\,d_{31} - d_{32}\,r_{31}}{{r_{31}}^2 + {r_{32}}^2} \end{cases} \qquad \text{(Eq.3.40)}$$

Les relations exprimant les angles duaux ϕ et ψ (Eq.3.39) permettent de conclure que cette méthode présente certaines singularités lorsque la quantité duale $sin\,\theta$ n'est pas inversible, c'est à dire si $Re(sin\,\theta) = 0$. En effet, la solution dégénère lorsque:

$$Re(\theta) = \quad \beta = \quad \begin{bmatrix} 0 \\ \text{ou} \\ \pi \end{bmatrix} \qquad \text{(Eq.3.41)}$$

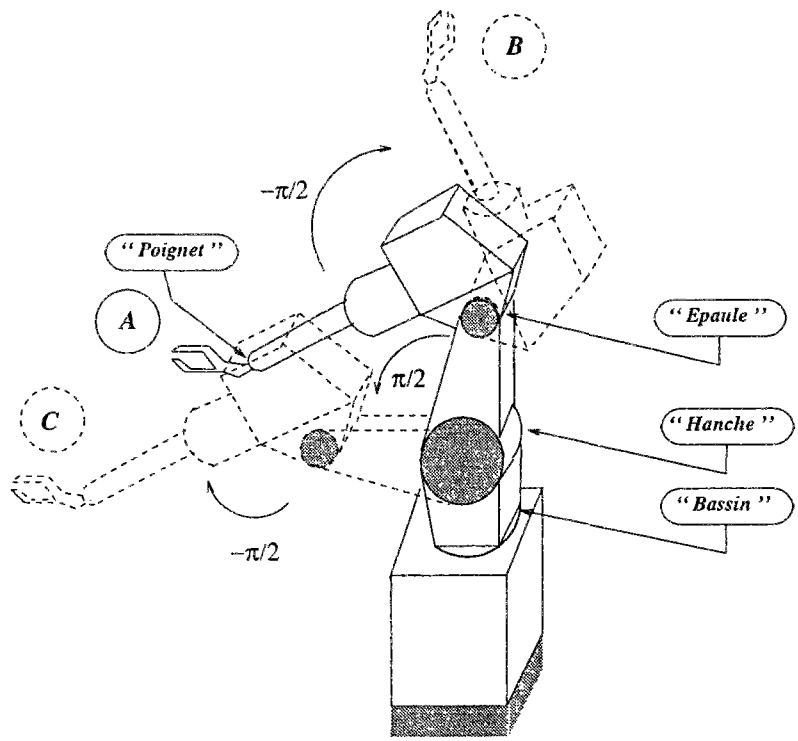
auquel cas les deux transformations intermédiaires se retrouvent colinéaires et le déplacement total se résume à un déplacement cylindrique d'axe Λ_ζ. Le déplacement $\boldsymbol{D}$ s'exprime, dans de tels conditions, par:

$$\boldsymbol{D} = \quad exp\,((\psi \pm \phi)\,\zeta) \qquad \text{(Eq.3.42)}$$

et l'ensemble des configurations singulières est donné par les valeurs de la nutation duale θ dans l'ensemble $\varepsilon\,\boldsymbol{R} \cup \pi \oplus \varepsilon\,\boldsymbol{R}$.

3.9 Modèle géométrique des mécanismes à structure ouverte simple

La configuration de l'organe terminal est obtenue en positionnant ce dernier par une succession de mouvements aux niveaux des articulations du bras manipulateur. Pour le moment nous allons considérer seulement des robots sous forme de chaînes ouvertes simples. Le modèle géométrique des chaînes cinématiques arborescentes sera donnée plus tard.



Le positionnement de l'organe terminal peut être décrit par un changement de configuration articulaire. Un changement de configuration articulaire se traduit alors par des déplacements élémentaires au niveaux des articulations du robot manipulateur. Ainsi:

- La configuration articulaire (B) est obtenue à partir de la configuration articulaire (A) par simple rotation d'un angle de $-\pi/2$ autour de l'articulation "Epaule".
- La configuration articulaire (C) est obtenue à partir de la configuration articulaire (A) par simple rotation d'un angle de $\pi/2$ autour de l'articulation "Hanche" et d'un angle de $-\pi/2$ autour de l'articulation "Epaule".

Fig. 3.16: Configuration articulaire

3.9.1 Déplacements de passage entre éléments adjacents du robot

Considérons un manipulateur en chaîne ouverte simple comportant N articulations. Nous allons donner au socle l'indice 0 et numéroter successivement les N éléments du robot, dans l'ordre où ils sont rencontrés lors du parcours de la chaîne cinématique, en partant de l'élément qui est directement lié au socle jusqu'à l'organe terminal qui portera alors l'indice N. Cette chaîne cinématique se compose donc de $N+1$ éléments $\mathcal{C}_0$, $\mathcal{C}_1$, ..., $\mathcal{C}_N$. Ensuite, pour numéroter les articulations nous allons associer le même indice à chaque élément et son articulation amont. Enfin, nous allons associer à chacun des corps $\mathcal{C}_i$ une famille fondamentale $\mathfrak{f}_i = (\xi_i, \eta_i, \zeta_i)$ qui en soit rigidement liée (cela revient au même d'associer aux corps $\mathcal{C}_i$ les repères orthonormés qui matérialisent les familles $\mathfrak{f}_i$). Notons $\boldsymbol{D}_{i+1}^{i}$ le déplacement de passage de la famille $\mathfrak{f}_i = (\xi_i, \eta_i, \zeta_i)$ à $\mathfrak{f}_{i+1} = (\xi_{i+1}, \eta_{i+1}, \zeta_{i+1})$. Si l'on se reporte à la figure Fig. 3.17 [Page 89]:

$$\mathfrak{f}_{i+1} = \boldsymbol{D}_{i+1}^{i} \diamond \mathfrak{f}_i \qquad \text{(Eq.3.43)}$$

Le déplacement inverse permet le passage de $\mathfrak{f}_{i+1} = (\xi_{i+1}, \eta_{i+1}, \zeta_{i+1})$ à $\mathfrak{f}_i = (\xi_i, \eta_i, \zeta_i)$, il sera noté dans la suite $\boldsymbol{D}_{i}^{i+1}$. Ainsi:

$$\boldsymbol{D}_{i}^{i+1} = \boldsymbol{D}_{i+1}^{i}{}^{-1} \qquad \text{(Eq.3.44)}$$

3.9.2 Modèle géométrique en terme de déplacements de passage

Si i et j sont deux indices relatifs à deux éléments du robot tels que $i > j$; alors d'après la relation (Eq.3.43) et la définition de l'opération "$\diamond$", le déplacement de passage entre les familles $\mathfrak{f}_i$ et $\mathfrak{f}_j$ peut être obtenu à partir de la relation récurrente:

$$\boldsymbol{D}_{j}^{i} = \boldsymbol{D}_{i-1}^{i}\, \boldsymbol{D}_{j}^{i-1} \qquad \text{(Eq.3.45)}$$

soit sous forme explicite:

$$\boldsymbol{D}_{j}^{i} = \boldsymbol{D}_{i-1}^{i}\, \boldsymbol{D}_{i-2}^{i-1} \cdots \boldsymbol{D}_{j}^{j+1} \qquad \text{(Eq.3.46)}$$

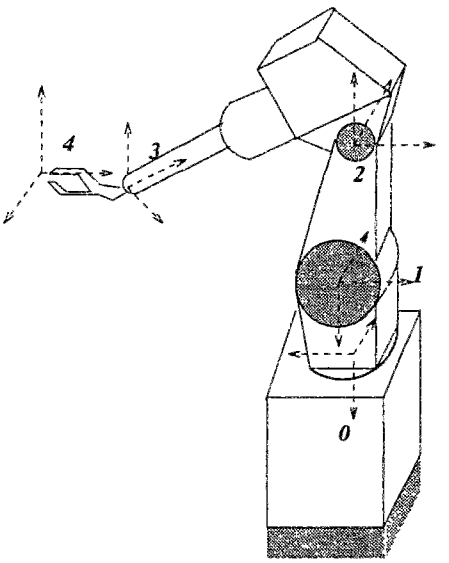
Donc, le déplacement de passage total permettant de décrire les configurations de l'organe terminal relativement à la famille $\mathfrak{f}_0$, liée au socle, est donné par:

$$\boldsymbol{D}_{0}^{N} = \boldsymbol{D}_{N-1}^{N}\, \boldsymbol{D}_{N-2}^{N-1} \cdots \boldsymbol{D}_{1}^{2}\, \boldsymbol{D}_{0}^{1} \qquad \text{(Eq.3.47)}$$

sont associés à des translations. Donc, la relation (??) est vérifiée dans le cas $N = 2$.

3.10 Conventions de Denavit–Hartenberg

La méthode des conventions de Denavit–Hartenberg [J. Denavit & R.S. Hartenberg [DENA-55]] est une démarche très répandue pour la description systématique du modèle géométrique des robots et mécanismes. Elle consiste à affecter des trièdres particuliers aux éléments du robot et ensuite à écrire le modèle géométriques en fonction des paramètres caractérisant



Le manipulateur ci-dessus se compose de 4 articulation. La chaîne robotique se compose donc de 5 éléments $\mathcal{C}_0$, $\mathcal{C}_1$, ..., $\mathcal{C}_4$. Les familles fondamentales qui en sont rigidement solidaires peuvent, à priori, être prises de façon arbitraires. On générale pour la résolution des problèmes d'ordre pratique, le choix des familles $\mathfrak{f}_i$ est fait de façon à ce qu'il simplifie les expression des quantités exprimés (coordonnées articulaires, générateur infinitésimaux des mouvements articulaires, ...).

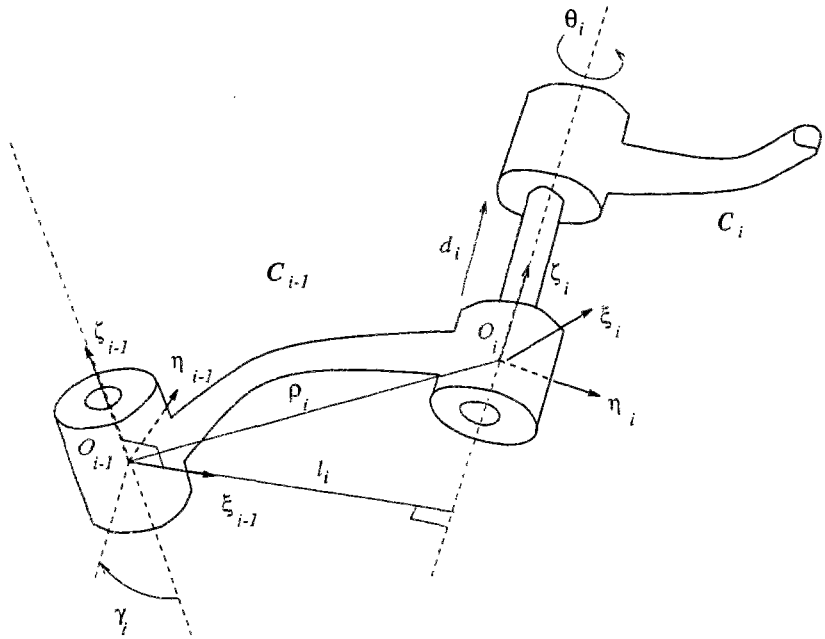
Fig. 3.17: Repérage des éléments du robot

les transformations entre les repères ainsi construits. Cette méthode est généralement présentée par une description matricielle d'ordre 4×4. La version que nous proposons ici est basée sur la théorie des groupes. En effet, nous avons adapté la méthode des notations de Denavit–Hartenberg au contexte général de la formulation de la mécanique par la théorie des groupes et algèbres de Lie en utilisant la notion des nombres duaux [A. Hamlili[HAML-91]]. Cette présentation conduit alors à une présentation matricielle d'ordre[9] 3×3.

3.10.1 Algorithme pour les systèmes en chaîne ouverte simple

(a) <u>Choix de la famille fondamentale associée au socle</u>: on affecte au socle fixe une famille fondamentale $\mathfrak{f}_0 = (\xi_0, \eta_0, \zeta_0)$, à priori, quelconque.

[9]La présentation matricielle duale 3×3 conduit à une présentation matricielle 3×3 dans R en séparant les parties réelles et duales des matrices duales.



Cette figure schématise la construction géométrique d'une famille associée à un élément du robot à partir de celle associée à son prédécesseur.

Fig. 3.18: Familles fondamentales associées aux notations de Denavit–Hartenberg

Initialisation et boucle de construction des familles $\mathfrak{f}_i$: pour i variant de 1 à N, on applique les étapes de (b1) à (b4):

(b1) **Construction du glisseur ζ_i**: ζ_i sera construit de façon que l'axe Λ_{ζ_i} coïncide avec l'axe du mouvement relatif de l'articulation i. Ainsi, si X_i est le générateur infinitésimal du mouvement relatif de l'articulation d'indice i, ζ_i est tel que:

$$\begin{cases} \zeta_i = \dfrac{X_i}{\| X_i \|_{\mathfrak{d}}} & \text{si } X_i \in \mathfrak{d} \setminus \mathfrak{t}, \\ \omega_{\zeta_i} = \dfrac{\vec{u}}{\| \vec{u} \|} & \text{si } X_i = \varepsilon \vec{u}. \end{cases} \qquad \text{(Eq.3.48)}$$

(b2) **Choix de l'origine de la famille $\mathfrak{f}_i$**:

- Si l'articulation d'indice i est prismatique (i.e. $Re\, X_i = \vec{0}$), l'origine O_i peut, à priori, être pris arbitrairement.
- si les axes $\Lambda_{\zeta_{i-1}}$ et Λ_{ζ_i} se coupent,l'origine O_i de la famille $\mathfrak{f}_i$ va pouvoir être pris à leur intersection,
- sinon, O_i sera pris à l'intersection de la normale commune aux axes $\Lambda_{\zeta_{i-1}}$ et Λ_{ζ_i} avec l'axe Λ_{ζ_i} [Fig. 3.18].

(b3) **Détermination du glisseur ξ_i**: si les axes $\Lambda_{\zeta_{i-1}}$ et Λ_{ζ_i} ne sont pas parallèles, le glisseur ξ sera pris tout simplement tel que:

$$\omega_{\xi_i} = \pm \frac{\omega_{\zeta_{i-1}} \times \omega_{\zeta_i}}{\| \omega_{\zeta_{i-1}} \times \omega_{\zeta_i} \|} \text{ et } \xi_i(M) = \omega_{\xi_i} \times \overrightarrow{O_i M}, \quad \forall M \in \mathcal{E} \qquad \text{(Eq.3.49)}$$

Sinon, on prendra ω_{ξ_i} un vecteur quelconque dans le plan perpendiculaire à Λ_{ζ_i} et:

$$\xi_i(M) = \omega_{\xi_i} \times \overrightarrow{O_i M}, \quad \forall M \in \mathcal{E} \qquad \text{(Eq.3.50)}$$

(b4) **Détermination du glisseur η_i**: on établit le glisseur η_i par la relation:

$$\eta_i = [\zeta_i, \xi_i] \qquad \text{(Eq.3.51)}$$

(c) **Identification des paramètres de Denavit–Hartenberg**: pour i variant de 1 à N on développe les étapes de (c1) à (c4):

(c1) **Détermination de l'angle γ_i**: γ_i est l'angle de rotation autour de l'axe Λ_{ξ_i} qui ramène l'axe $\Lambda_{\zeta_{i-1}}$ à une droite parallèle à Λ_{ζ_i}. Ainsi:

$$\gamma_i = arcos\left(\omega_{\zeta_{i-1}}.\omega_{\zeta_i}\right) \qquad \text{(Eq.3.52)}$$

(c2) **Détermination de la distance l_i**: l_i est la distance depuis l'origine O_i de la famille $\mathfrak{f}_i$ jusqu'au point d'intersection des axes $\Lambda_{\zeta_{i-1}}$ et Λ_{ξ_i} mesurée sur l'axe Λ_{ξ_i}. Ainsi:

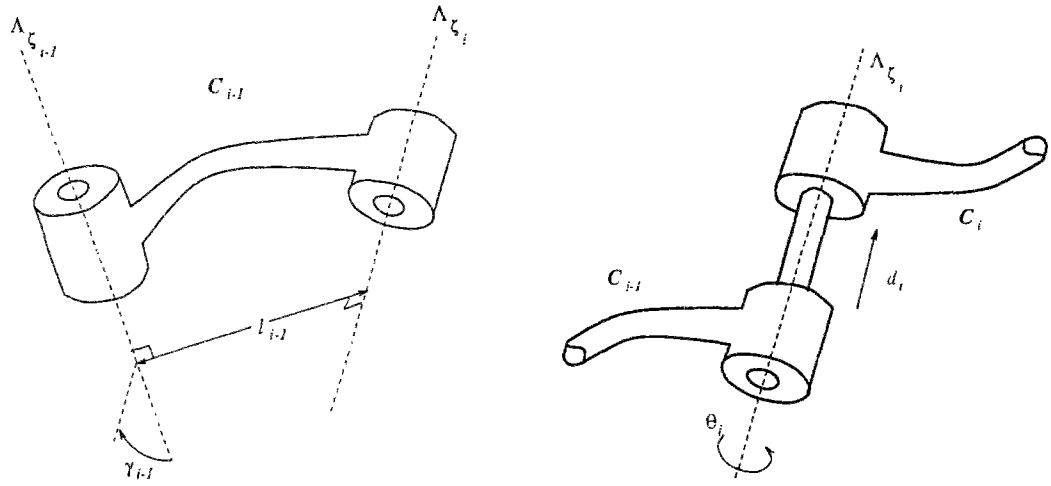
$$l_i = -\frac{[\zeta_{i-1}/\zeta_i]}{\sqrt{1-(\omega_{\zeta_{i-1}}.\omega_{\zeta_i})^2}} \qquad \text{(Eq.3.53)}$$

(c3) **Détermination de l'angle θ_i**: θ_i est l'angle de rotation autour de l'axe $\Lambda_{\zeta_{i-1}}$ qui ramène l'axe $\Lambda_{\xi_{i-1}}$ à une droite parallèle à Λ_{ξ_i}. Ainsi:

$$\theta_i = arcos\left(\omega_{\xi_{i-1}}.\omega_{\xi_i}\right) \qquad \text{(Eq.3.54)}$$

(c4) **Détermination de la distances d_i**: d_i est la distance mesurée sur l'axe $\Lambda_{\zeta_{i-1}}$ entre l'origine O_{i-1} de la famille $\mathfrak{f}_{i-1}$ à l'intersection des axes $\Lambda_{\zeta_{i-1}}$ et Λ_{ξ_i}. Ainsi:

$$d_i = -\frac{[\xi_{i-1}/\xi_i]}{\sqrt{1-(\omega_{\xi_{i-1}}.\omega_{\xi_i})^2}} \qquad \text{(Eq.3.55)}$$



Les quatre paramètres associés aux notations de Denavit–Hartenberg peuvent être regrouper en deux classes:

- Deux paramètres constants qui caractérisent la disposition géométrique du corps du même indice par rapport à la disposition de son prédécesseur.
- Deux paramètres variables qui caractérisent le mouvement relatifs au niveau de l'articulation du même indice.

Sur la figure de gauche sont représentés les paramètres géométriques constants et sur celle de droite sont représentés les paramètres cinématiques variables.

Fig. 3.19: Paramètres associés au notations de Denavit–Hartenberg

(d) **Formation des déplacements D_i^{i-1} relatifs aux familles $\mathfrak{f}_i$**: pour i variant de 1 à N, le déplacement D_{i-1}^{i} relatif à l'articulation i est obtenu par la relation:

$$D_i^{i-1} = B_i A_i^{i-1} \text{ où } A_i^{i-1} = exp(\alpha_i \zeta_{i-1}) \text{ et } B_i = exp(\beta_i \xi_i) \qquad \text{(Eq.3.56)}$$

avec:

$$\alpha_i = \theta_i + \varepsilon d_i \text{ et } \beta_i = \gamma_i + \varepsilon l_i \qquad \text{(Eq.3.57)}$$

(e) **Elaboration des matrices duales relatives aux opérateurs $D_i^{i-1}{}_*$**: pour i variant de 1 à N, la matrice qui correspond à la représentation adjointe du déplacement D_{i-1}^{i},

relativement à la famille $\mathfrak{f}_i$ est donnée par:

$$[\boldsymbol{D}_i^{i-1}{}_*] = [\boldsymbol{B}_{i*}]\,[\boldsymbol{A}_i^{i-1}{}_*]$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\beta_i & -sin\,\beta_i \\ 0 & sin\,\beta_i & cos\,\beta_i \end{pmatrix} \begin{pmatrix} cos\,\alpha_i & -sin\,\alpha_i & 0 \\ sin\,\alpha_i & cos\,\alpha_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{(Eq.3.58)}$$

i.e.

$$[\boldsymbol{D}_i^{i-1}{}_*] = \begin{pmatrix} cos\,\alpha_i & -sin\,\alpha_i & 0 \\ cos\,\beta_i\,sin\,\alpha_i & cos\,\beta_i\,cos\,\alpha_i & -sin\,\beta_i \\ sin\,\beta_i\,sin\,\alpha_i & sin\,\beta_i\,cos\,\alpha_i & cos\,\beta_i \end{pmatrix} \qquad \text{(Eq.3.59)}$$

3.10.2 Commentaires et preuve de l'algorithme

L'intérêt du langage mathématique que nous avons proposé dans cet algorithme réside dans le fait que la détermination des paramètres de Denavit–Hartenberg est effectuée par une méthode analytique très simple [Relations (Eq.3.52), (Eq.3.53), (Eq.3.54) et (Eq.3.55)]. Désormais, l'automatisation du processus de calcul des paramètres de Denavit–Hartenberg sera possible dès que les familles fondamentales (i.e. les repères) associées aux éléments du robot sont données. En résumé l'algorithme général que nous venons de présenter se compose de trois parties:

- **Phase 1**: la construction des familles fondamentales associées aux éléments du robot comprend les étapes (a) et (b), où (b) est une boucle itérant les étapes de (b1) à (b4).
- **Phase 2**: l'identification des paramètres de Denavit–Hartenberg est donnée par la boucle (c) itérant les étapes de (c1) à (c4).
- **Phase 3**: la construction des matrices duales de passage entre les familles fondamentales définies dans la phase 1. Cette phase comprend deux boucles: (d) au terme de quoi sont formés les déplacements de passage et la boucle (e) qui permet de calculer les matrices de passage associées à ces déplacements.

Par ailleurs, seules les relations (Eq.3.52), (Eq.3.53), (Eq.3.54) et (Eq.3.55) nécessitent une justification complémentaire. Pour ce faire, il suffit de voir que $\{\xi_{i-1}/\xi_i\}$ et $\{\zeta_{i-1}/\zeta_i\}$ sont les cosinus directeurs duaux définis par le produit scalaire dual des torseurs ξ_{i-1} par ξ_{i-1} d'une part et de ζ_{i-1} par ζ_i d'autre part [le paragraphe 3.7, page 75]. Or, la définition géométrique des paramètres γ_i, l_i, θ_i et d_i permet d'écrire:

$$\{\xi_{i-1}/\xi_i\} = cos\,(\theta_i + \varepsilon\, d_i) \text{ et } \{\zeta_{i-1}/\zeta_i\} = cos\,(\gamma_i + \varepsilon\, l_i)$$

où *cos* est le prolongement canonique dual de la fonction cosinus. De plus, d'après le théorème 1.7 [Chapitre 1, page 21] on peut écrire:

$$\theta_i + \varepsilon\, d_i = arcos\,(\{\xi_{i-1}/\xi_i\}) \text{ et } \gamma_i + \varepsilon\, l_i = arcos\,(\{\zeta_{i-1}/\zeta_i\}) \qquad \text{(Eq.3.60)}$$

où *arccos* est le prolongement canonique dual de la fonction arc cosinus. Ce passage est justifié par le théorème 1.7 [Chapitre 1, page 21]. Les relations (Eq.3.52), (Eq.3.53), (Eq.3.54) et (Eq.3.55) en

découlent.

Par ailleurs, la relation (Eq.3.59) implique, compte tenu des relations (Eq.3.57):

$$[\boldsymbol{D}^i_{i-1\,*}] = \begin{pmatrix} cos\,\alpha_i & cos\,\beta_i\,sin\,\alpha_i & sin\,\beta_i\,sin\,\alpha_i \\ -sin\,\alpha_i & cos\,\beta_i\,cos\,\alpha_i & sin\,\beta_i\,cos\,\alpha_i \\ 0 & -sin\,\beta_i & cos\,\beta_i \end{pmatrix}$$

le calcul des parties réelle et duale de la matrice $[\boldsymbol{D}^i_{i-1\,*}]$ donne:

$$Re\,[\boldsymbol{D}^i_{i-1\,*}] = \begin{pmatrix} cos\,\theta_i & cos\,\gamma_i\,sin\,\theta_i & sin\,\gamma_i\,sin\,\theta_i \\ -sin\,\theta_i & cos\,\gamma_i\,cos\,\theta_i & sin\,\gamma_i\,cos\,\theta_i \\ 0 & -sin\,\gamma_i & cos\,\gamma_i \end{pmatrix} \qquad \text{(Eq.3.61)}$$

et

$$Du\,[\boldsymbol{D}^i_{i-1\,*}] = \begin{pmatrix} -d_i\,sin\,\theta_i & cos\,\gamma_i\,d_i\,cos\,\theta_i - l_i\,sin\,\gamma_i\,sin\,\theta_i & sin\,\gamma_i\,d_i\,cos\,\theta_i + l_i\,cos\,\gamma_i\,sin\,\theta_i \\ -d_i\,cos\,\theta_i & -cos\,\gamma_i\,d_i\,sin\,\theta_i - l_i\,sin\,\gamma_i\,cos\,\theta_i & -sin\,\gamma_i\,d_i\,sin\,\theta_i + l_i\,cos\,\gamma_i\,cos\,\theta_i \\ 0 & -l_i\,cos\,\gamma_i & -l_i\,sin\,\gamma_i \end{pmatrix}$$

3.11 Modèle géométrique d'un manipulateur à structure arborescente

Dans cette section, nous allons décrire les aspects algorithmiques permettant d'obtenir le modèle géométrique associé à des systèmes articulés à topologie arborescente, mais nous conservons, néanmoins, la terminologie définie pour les structures à chaîne simple et les notations définies dans la section 3.5 pour les systèmes arborescents.

3.11.1 Formulation itérative du modèle géométrique

Considérons un système articulé à structure arborescente dont les éléments sont numérotés selon un ordre récursivement reconnaissable sur le graphe du mécanisme. Si j et k sont les indices de deux éléments d'une même branche tels que $j < k$ et si $i^-(k)$ définit l'indice du prédécesseur immédiat du corps d'indice k, alors, la traduction de la relation récurrente (Eq.3.45) pour un tel système s'écrit:

$$\boldsymbol{D}^k_j = \boldsymbol{D}^k_{i^-(k)}\,\boldsymbol{D}^{i^-(k)}_j \qquad \text{(Eq.3.62)}$$

Donc, si la structure arborescente possède K organes terminaux d'indices respectifs $N_1, N_2, \cdots, N_K$, le modèle géométrique est obtenu en calculant $\boldsymbol{D}^{N_1}_0, \boldsymbol{D}^{N_2}_0, \cdots, \boldsymbol{D}^{N_K}_0$ selon le modèle récurrent (Eq.3.62).

Exemple

Reprenons l'exemple du robot à structure arborescente de la figure [Fig. 3.11, page 73]. Ce robot possède trois organes terminaux, donc, son modèle géométrique est donnée par les trois

déplacements:

$$\begin{cases} \boldsymbol{D}_0^8 = \boldsymbol{D}_5^8\,\boldsymbol{D}_3^5\,\boldsymbol{D}_1^3\boldsymbol{D}_0^1 \\ \boldsymbol{D}_0^7 = \boldsymbol{D}_4^7\,\boldsymbol{D}_2^4\,\boldsymbol{D}_1^2\boldsymbol{D}_0^1 \\ \boldsymbol{D}_0^6 = \boldsymbol{D}_4^6\,\boldsymbol{D}_2^4\,\boldsymbol{D}_1^2\boldsymbol{D}_0^1 \end{cases}$$

3.11.2 Généralisation de l'algorithme par les conventions de Denavit–Hartenberg aux systèmes arborescents

(a) **Choix de la famille fondamentale associée au socle**: on affecte au socle fixe une famille fondamentale $\mathfrak{f}_0 = (\xi_0, \eta_0, \zeta_0)$, à priori, quelconque.

(b) **Initialisation et boucle de construction des familles $\mathfrak{f}_k$**: pour k variant de 1 à N et $j = i^-(k)$, on applique les étapes de (b1) à (b4):

(b1) **Construction du glisseur ζ_k**: ζ_k sera construit de façon que l'axe Λ_{ζ_k} coïncide avec l'axe du mouvement relatif de l'articulation k. Ainsi, si X_k est le générateur infinitésimal du mouvement relatif de l'articulation d'indice k, ζ_k est tel que:

$$\begin{cases} \zeta_k = \dfrac{X_k}{\| X_k \|_0} & \text{si } X_k \in \mathfrak{d} \setminus \mathfrak{t}, \\ \omega_{\zeta_k} = \dfrac{\vec{u}}{\| \vec{u} \|} & \text{si } X_k = \varepsilon\,\vec{u}. \end{cases}$$

(b2) **Choix de l'origine de la famille $\mathfrak{f}_k$**:

- Si l'articulation d'indice k est prismatique (i.e. $Re\,X_k = \vec{0}$), l'origine O_k peut, à priori, être pris arbitrairement.
- si les axes Λ_{ζ_j} et Λ_{ζ_k} se coupent,l'origine O_k de la famille $\mathfrak{f}_k$ va pouvoir être pris à leur intersection,
- sinon, O_k sera pris à l'intersection de la normale commune aux axes Λ_{ζ_j} et Λ_{ζ_k} avec l'axe Λ_{ζ_k} [Fig. 3.20].

(b3) **Détermination du glisseur ξ_k**: si les axes Λ_{ζ_j} et Λ_{ζ_k} ne sont pas parallèles, le glisseur ξ sera pris tout simplement tel que:

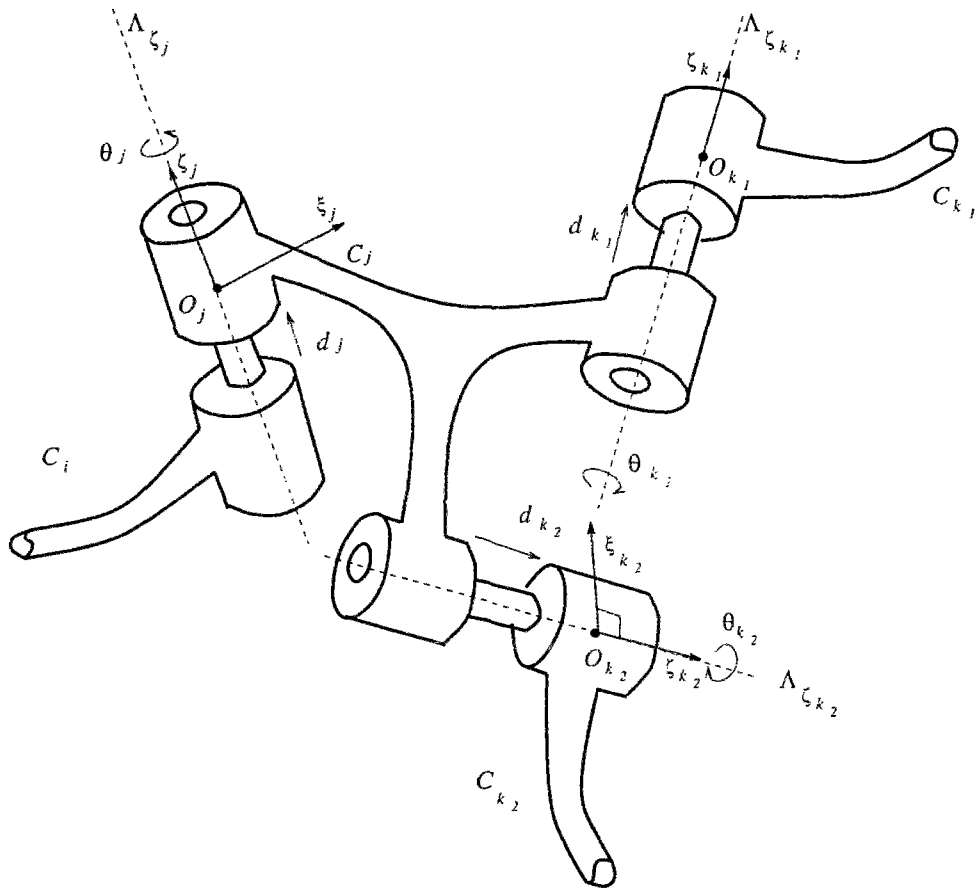
$$\omega_{\xi_k} = \pm\frac{\omega_{\zeta_j} \times \omega_{\zeta_k}}{\| \omega_{\zeta_j} \times \omega_{\zeta_k} \|} \text{ et } \xi_k(M) = \omega_{\xi_k} \times \overrightarrow{O_k M}, \quad \forall M \in \mathcal{E}$$

Sinon, on prendra ω_{ξ_k} un vecteur quelconque dans le plan perpendiculaire à Λ_{ζ_k} et:

$$\xi_k(M) = \omega_{\xi_k} \times \overrightarrow{O_k M}, \quad \forall M \in \mathcal{E}$$

(b4) **Détermination du glisseur η_k**: on établit le glisseur η_k par la relation:

$$\eta_k = [\zeta_k, \xi_k]$$



Cette figure schématise les paramètres cinématique associés aux éléments adjacents d'un système à structure arborescente de corps articulés indéformables.

Fig. 3.20: Paramètres cinématiques pour un système arborescent

(c) **Identification des paramètres de Denavit–Hartenberg**: pour k variant de 1 à N et $j = i^-(k)$, on développe les étapes de (c1) à (c4):

(c1) **Détermination de l'angle γ_k**: γ_k est l'angle de rotation autour de l'axe Λ_{ξ_k} qui ramène l'axe Λ_{ζ_j} à une droite parallèle à Λ_{ζ_k}. Ainsi:

$$\gamma_k = \arccos\left(\omega_{\zeta_j}.\omega_{\zeta_k}\right)$$

(c2) **Détermination de la distance l_k**: l_k est la distance depuis l'origine O_k de la famille $\mathfrak{f}_k$ jusqu'au point d'intersection des axes Λ_{ζ_j} et Λ_{ξ_k} mesurée sur l'axe Λ_{ξ_k}. Ainsi:

$$l_k = -\frac{[\zeta_j/\zeta_k]}{\sqrt{1-(\omega_{\zeta_j}.\omega_{\zeta_k})^2}}$$

(c3) **Détermination de l'angle θ_k**: θ_k est l'angle de rotation autour de l'axe Λ_{ζ_j} qui ramène l'axe Λ_{ξ_j} à une droite parallèle à Λ_{ξ_k}. Ainsi:

$$\theta_k = arcos\,(\omega_{\xi_j}.\omega_{\xi_k})$$

(c4) **Détermination de la distances d_k**: d_k est la distance mesurée sur l'axe Λ_{ζ_j} entre l'origine O_j de la famille $\mathfrak{f}_j$ à l'intersection des axes Λ_{ζ_j} et Λ_{ξ_k}. Ainsi:

$$d_k = -\frac{[\xi_j/\xi_k]}{\sqrt{1-(\omega_{\xi_j}.\omega_{\xi_k})^2}}$$

(d) **Formation des déplacements $\boldsymbol{D}_k^j$ relatifs aux familles $\mathfrak{f}_k$**: pour k variant de 1 à N et $j = i^-(k)$, le déplacement $\boldsymbol{D}_j^k$ relatif à l'articulation d'indice k est obtenu par la relation:

$$\boldsymbol{D}_k^j = \boldsymbol{B}_k\,\boldsymbol{A}_k^j \text{ où } \boldsymbol{A}_k^j = exp(\alpha_k\,\zeta_j) \text{ et } \boldsymbol{B}_k = exp(\beta_k\,\xi_k)$$

avec: $\alpha_k = \theta_k + \varepsilon\,d_k$ et $\beta_k = \gamma_k + \varepsilon\,l_k$.

(e) **Elaboration des matrices duales relatives aux opérateurs $\boldsymbol{D}_{k*}^j$**: pour k variant de 1 à N et $j = i^-(k)$: $\left[\boldsymbol{D}_{k*}^j\right] = [\boldsymbol{B}_{k*}]\,\left[\boldsymbol{A}_{k*}^j\right]$.

3.12 Conclusion

Les méthodes que nous avons développées dans ce chapitre s'appliquent à un grand nombre de mécanismes. L'application de la théorie des groupes à l'ensemble des déplacements et à la caractérisation des liaisons mécaniques offre un langage naturel intéressant pour la description du modèle géométrique des mécanismes. Ce langage favorise en particulier une formulation analytique et un calcul symbolique simple des modèles du comportement géométrique et cinématique des systèmes mécaniques articulés (i.e. les expressions syntaxiques des modèles dans un tel langage s'expriment de façon très simple). Ces méthodes peuvent donc être implémentées dans un processus de calcul formel. La théorie des graphes offre alors l'outil pour généraliser ces méthodes et modèles à l'étude de systèmes articulés plus complexes. On a vu que pour déterminer les indicatrices de la structure arborescente ouverte, il suffit de connaître la matrice des prédécesseurs immédiats. Cette méthode est très économique par rapport à celle utilisée par Wittenburg [WITT-75] qui pour déterminer les indicatrices[10] de la structure arborescente. Dans les développements de Wittenburg [WITT-75],

[10]Les indicatrices définies par Wittenburg dans son exposé sont, en fait, différentes des indicatrices que nous avons définies dans le cadre de notre travail.

la détermination des indicatrices nécessite la connaissance des matrices d'incidence et de référence de l'arbre topologique. Cependant, la connaissance complète de ces deux matrices introduit une information inutile pour le calcul itératif du modèle dynamique. En effet, la reconnaissance du graphe dans un processus itératif va se faire de proche en proche et par la suite l'on aura seulement besoin d'une description locale du graphe et non une connaissance globale. Dans le chapitre suivant nous appliquerons ces techniques et résultats au calcul du modèle dynamique des mécanismes.

Chapitre 4

MODELE DYNAMIQUE DES SYSTEMES ARTICULES

4.1 Introduction

Jusqu'à présent, notre analyse des systèmes articulés était uniquement focalisée sur des considérations cinématiques. Dans ce chapitre, nous présentons les causes qui sont à l'origine du mouvement de tels mécanismes. Les buts que nous nous fixons dans ce chapitre peuvent être résumés en trois points essentiels:

- Apporter une contribution à la modélisation du problème dynamique à partir du formalisme des groupes et algèbres de Lie et des outils de modélisation que nous avons développés jusqu'ici avec la notion de nombres duaux.
- Automatiser le processus de génération des équations du mouvement des systèmes mécaniques articulés. Pour ce faire, nous allons établir une formulation itérative quasi-optimale permettant de calculer le modèle dynamique d'un système articulé à chaîne ouverte simple ou arborescente.
- Relier les modèles dans le formalisme des groupes et algèbres de Lie avec les modèles classiques de la mécanique.

Comme nous allons le voir, une comparaison selon l'efficacité du calcul algorithmique, entre notre formulation et sept autres méthodes de référence parmi les algorithmes les plus réputés pour leur efficacité, permettra de mettre en valeur la puissance de notre formulation et son adaptabilité à la commande dynamique des robots en temps réel. Comme nous allons le voir dans un prochain chapitre, les performances de nos algorithmes peuvent atteindre une efficacité optimale grâce:

- d'une part, à une implémentation itérative symbolique utilisant l'outil de calcul formel et les techniques de réécriture pour le regroupement des termes qui interviennent plusieurs fois dans le schéma des calculs,

- d'autre part, à l'automatisation du processus de simplification de tels calculs chaque fois qu'interviennent des termes nuls ou des multiplications par ±1 [Chapitre 6, section 6.5, page 183].

4.2 Descriptions du champ des vitesses

4.2.1 Dérivée temporelle d'un champ

Considérons un champ de vecteurs Y variable au cours du temps (i.e. $t \mapsto Y(t)$ définit une application de $\boldsymbol{R}$ dans $\mathfrak{D}$). Si on se fixe une famille fondamentale $\mathfrak{f} = (\xi, \eta, \zeta)$ de l'algèbre de Lie des torseurs $\mathfrak{D}$, on peut exprimer Y par son système de coordonnées par rapport à la famille fondamentale $\mathfrak{f}$:

$$Y = (\alpha + \varepsilon\, a)\,\xi + (\beta + \varepsilon\, b)\,\eta + (\gamma + \varepsilon\, c)\,\zeta$$

la dérivée temporelle du champ de vecteur Y, rapporté à la famille fondamentale $\mathfrak{f}$, est complètement déterminée par les dérivées des coordonnées de Y relativement à la famille $\mathfrak{f}$; c'est à dire:

$$\frac{d}{dt}Y = (\dot{\alpha} + \varepsilon\, \dot{a})\,\xi + (\dot{\beta} + \varepsilon\, \dot{b})\,\eta + (\dot{\gamma} + \varepsilon\, \dot{c})\,\zeta \qquad \text{(Eq.4.1)}$$

Remarquons cependant que si M est un point mobile: $\dfrac{d}{dt}[Y(M)] \neq \dot{Y}(M)$.

4.2.2 Descriptions eulerienne et lagrangienne des vitesses

Généralement en mécanique du corps solide on définit deux descriptions du mouvement selon que l'on se réfère à l'espace ou au solide lui même.

4.2.2.a Champ eulerien de vitesse

Définition 4.1

Le champ eulerien de vitesse (ou tout simplement vitesse eulerienne) est le champ des vitesses du corps dans un référentiel lié à l'espace.

Pour un mouvement de corps rigide caractérisé par $t \mapsto \boldsymbol{D}(t)$ relativement à une configuration de référence $\boldsymbol{r}$, la théorie des groupes et algèbres de Lie permet d'exprimer le champ eulerien de vitesse à partir de la dérivée à droite du déplacement $\boldsymbol{D}$ par rapport à la variable temporelle:

$$U = R_{D^{-1}}^{T} \frac{d\,\boldsymbol{D}}{dt} \qquad \text{(Eq.4.2)}$$

4.2.2.b Champ des vitesses lagrangiennes

Définition 4.2

Le champ lagrangien de vitesse (ou tout simplement vitesse lagrangienne) est le champ des vitesses du solide dans un référentiel lié au solide lui même.

Soit un mouvement de corps rigide caractérisé par l'application $t \mapsto \boldsymbol{D}(t)$ relativement à une configuration de référence $\boldsymbol{r}$. D'après la théorie générale des groupes et algèbres de Lie, le champ

lagrangien de vitesse peut être exprimé à partir de la dérivée à gauche du déplacement $\boldsymbol{D}$ par rapport à la variable temporelle. Ainsi:

$$V = L^T_{D^{-1}} \frac{d\boldsymbol{D}}{dt} \qquad \text{(Eq.4.3)}$$

Corollaire 4.1

Si U et V désignent respectivement les champs eulerien et lagrangien de vitesse d'un mouvement donné par $t \mapsto \boldsymbol{D}(t)$ relativement à configuration de référence, alors:

$$V = \boldsymbol{D}_*^{-1} U \qquad \text{(Eq.4.4)}$$

4.2.2.c Cas du sous-groupe des mouvements cylindriques

Le cas des paires cinématiques de type C (paires cylindriques) présente un intérêt particulier dans la modélisation des systèmes mécaniques articulés. En effet, ce cas englobe toutes les articulations dont le mouvement relatif entre les deux éléments de la paire cinématique se réduit à un mouvement selon un axe fixe. Nous avons vu, au chapitre précédent, qu'un mouvement quelconque peut être décomposé en trois mouvements de types C. En fait, un mouvement de type C peut être caractérisé par le fait que le générateur infinitésimal du déplacement $\boldsymbol{D}(t) = exp X(t)$, relativement à une configuration de référence, peut se mettre sous la forme:

$$X(t) = \alpha(t)\zeta \text{ avec } \alpha \in C^2(\boldsymbol{R}, \Delta) \text{ et } \zeta \in \mathfrak{D} \text{ telle que } \| \zeta \|_{\mathfrak{d}} = 1 \qquad \text{(Eq.4.5)}$$

où $C^2(\boldsymbol{R}, \Delta)$ est l'ensemble des fonctions d'une variable réelle et à valeurs duales deux fois continuement dérivables. Cette condition de dérivabilité traduit dans ce cas particulier le principe de dérivabilité de la cinématique (i.e. existence des vitesses et des accélérations). Les mouvements de type C recouvrent toutes les paires cinématiques associées à des sous-groupes de C (les déplacements cinématiquement admissibles par les articulations de type C forment un sous-groupe du groupe de Lie $\mathbb{D}$ -qui admet à son tours des sous-groupes propres- et ,en tant que variété différentielle sur $\boldsymbol{R}$, il est de dimension 2). On retrouve ainsi les quatre types d'articulations mécaniques à axe fixe (pour le mouvement relatif) au cours du temps:

- les paires cinématiques prismatiques (un degré de liberté en translation) de type P,
- les paires cinématiques pivots (un degré de liberté en rotation) de type R,
- les paires cinématiques hélicoïdales (un degré de liberté suivant un axe) de type H,
- les paires cinématiques cylindriques (deux degrés de liberté: une rotation et une translation indépendantes suivant le même axe) de type C.

Corollaire 4.2

Si $t \mapsto \boldsymbol{D}(t) = exp X(t)$ désigne un mouvement de type C, alors le champ des vitesses euleriennes et confondu avec le champ des vitesses lagrangiennes et on a:

$$X = \alpha\zeta \Rightarrow U = V = \dot{\alpha}\zeta \qquad \text{(Eq.4.6)}$$

4.3 Accélérations d'un corps rigide – Lois de composition de mouvements

Avant d'entamer la description des accélérations, nous allons tout d'abord discuter brièvement le problème de différentiation de la représentation adjointe du déplacement associé à un mouvement donné.

4.3.1 Résultats préliminaires

Lemme 4.3

Considérons l'application $t \mapsto Y(t)$ de $\boldsymbol{R}$ dans $\mathfrak{d}$ et l'application $t \mapsto \boldsymbol{D}(t)$ définissant un mouvement donné par rapport à une configuration de référence. Si V désigne la vitesse lagrangienne de ce mouvement et si on pose $Z(t) = \boldsymbol{D}_* Y(t)$, alors:

$$\dot{Y} = \boldsymbol{D}_*^{-1} \dot{Z} + [\boldsymbol{D}_*^{-1} Z, V] \qquad \text{(Eq.4.7)}$$

Preuve :

Soient O et N deux points fixes de $\mathcal{E}$, alors l'application $t \mapsto M(t) = \boldsymbol{D}(N)$ définit un point mobile $M(t)$ animé du mouvement correspondant à $t \mapsto \boldsymbol{D}(t)$. Rappelons d'abord la règle générale de dérivation:

$$\frac{dY(M)}{dt} = \dot{Y}(M) + \omega_Y \times \frac{d\overrightarrow{OM}}{dt} = \dot{Y}(M) + \omega_Y \times U(M) \qquad \text{(Eq.4.8)}$$

Maintenant, posons $Z(t) = \boldsymbol{D}_* Y(t)$ et notons D la partie linéaire $\boldsymbol{D}$, alors on peut écrire:

$$Z(M) = [D \circ Y](N) \qquad \text{(Eq.4.9)}$$

D'une part, compte tenu de (Eq.4.8) on a:

$$\frac{dZ(M)}{dt} = \dot{Z}(M) + \omega_Z \times U(M) \qquad \text{(Eq.4.10)}$$

et, d'autre part, en dérivant relation (Eq.4.9) par rapport à t on obtient:

$$\begin{aligned} \frac{dZ(M)}{dt} &= \omega_U \times D(Y(N)) + D(\dot{Y}(N)) \\ &= \omega_U \times [\boldsymbol{D}_* Y](\boldsymbol{D}(N)) + [\boldsymbol{D}_* \dot{Y}](\boldsymbol{D}(N)) \\ &= \omega_U \times Z(M) + [\boldsymbol{D}_* \dot{Y}](M) \end{aligned} \qquad \text{(Eq.4.11)}$$

les deux relations (Eq.4.10) et (Eq.4.11) sont équivalentes, donc:

$$\dot{Z}(M) = [\boldsymbol{D}_* \dot{Y}](M) + \omega_U \times Z(M) - \omega_Z \times U(M)$$

cela entraîne: $\dot{Z} = \boldsymbol{D}_* \dot{Y} + [U, Z]$, d'où la relation (Eq.4.7) compte tenu du corollaire 4.1. ∎

Théorème 4.4

Soit l'application $t \mapsto \boldsymbol{D}(t)$ décrivant un mouvement donnée par rapport à une configuration de référence. Si V désigne son champ des vitesses lagrangiennes , alors:

$$\frac{d}{dt} \boldsymbol{D}_* = \boldsymbol{D}_* \circ ad\, V \qquad \text{(Eq.4.12)}$$

et

$$\frac{d}{dt}\boldsymbol{D}_*^{-1} = -ad\,V \circ \boldsymbol{D}_*^{-1} \qquad \text{(Eq.4.13)}$$

Preuve :

La relation (Eq.4.13) est évidente. Il suffit de prendre Z un torseur fixe de $\mathfrak{d}$, considérer $Y(t) = \boldsymbol{D}_*^{-1} Z$ et d'appliquer le lemme 4.3.

La relation (Eq.4.12) n'est pas plus difficile à obtenir, il suffit d'appliquer le lemme 4.3 avec $Y \in \mathfrak{d}$ (i.e. Y ne dépend pas du temps) et le champ Z tel que $Z(t) = \boldsymbol{D}_* Y$. ∎

Corollaire 4.5

Soit U le champ des vitesses euleriennes associé au mouvement caractérisé $t \mapsto \boldsymbol{D}(t)$ relativement à une configuration de référence $\boldsymbol{r}$. Alors:

$$\frac{d}{dt}\boldsymbol{D}_* = ad\,U \circ \boldsymbol{D}_* \qquad \text{(Eq.4.14)}$$

et

$$\frac{d}{dt}\boldsymbol{D}_*^{-1} = -\boldsymbol{D}_*^{-1} \circ ad\,U \qquad \text{(Eq.4.15)}$$

Preuve :

Il suffit de voir que dans (Eq.4.12) et (Eq.4.13), $V = \boldsymbol{D}_*^{-1} U$. Les relations (Eq.4.14) et (Eq.4.15) en découlent directement. ∎

4.3.2 Vitesses et accélérations dans une paire cinématique

Soit une paire cinématique donnée $\mathbb{S}_{ij} = \{\mathcal{C}_i, \mathcal{C}_j\}$, notons $\mathfrak{f}_i$ et $\mathfrak{f}_j$ deux familles mobiles rigidement solidaires à $\mathcal{C}_i$ et $\mathcal{C}_j$ respectivement.

4.3.2.a Notations

Dans la suite nous adoptons la notation qui consiste à noter ${}^iX$ le système des coordonnées duales du torseur X relativement à la famille $\mathfrak{f}_i$ et ${}^jX$ ses coordonnées par rapport à la famille $\mathfrak{f}_j$. Ainsi, dans ces notations, $\boldsymbol{D}_j^i(t)$ représente le mouvement de la famille $\mathfrak{f}_i$ par rapport à $\mathfrak{f}_j$ et ${}^iV_i^j$ les coordonnées du torseur lagrangien de vitesse relative du corps $\mathcal{C}_i$ par rapport à $\mathcal{C}_j$ et enfin ${}^iV_i$ et ${}^jV_j$ les coordonnées des champs lagrangiens de vitesse des corps $\mathcal{C}_i$ et $\mathcal{C}_j$, respectivement rapportés aux familles $\mathfrak{f}_i$ et $\mathfrak{f}_j$.

4.3.2.b Composition des vitesses

La loi de composition des vitesses se traduit par le fait que la vitesse absolue d'un corps est égale à la somme de ses vitesses relatives et d'entraînement. Ainsi, pour les éléments de la paire cinématique $\mathbb{S}_{ij}$ cette loi s'exprime par la relation:

$${}^iV_i = \boldsymbol{D}_{j*}^i\,{}^jV_j + {}^iV_i^j \qquad \text{(Eq.4.16)}$$

Les algorithmes de calcul des vitesses angulaire et linéaire du corps C_i à partir de celles du corps C_j peuvent être retrouvés par l'extraction des parties réelle et duale pures de la représentation matricielle de la relation (Eq.4.16). Notons D_j^i la partie linéaire du déplacement $\boldsymbol{D}_j^i$, la loi de composition des vitesses angulaires s'exprime par:

$$ {}^i\omega_i = D_j^i \, {}^j\omega_j + {}^i\omega_i^j \qquad \text{(Eq.4.17)} $$

et la composition des vitesses linéaires se déduit de (Eq.4.16) par:

$$ \underbrace{{}^iV_i(M)}_{\text{vitesse absolue}} = \underbrace{D_j^i \, {}^jV_j(O_j) + D_j^i \, {}^j\omega_j \times D_j^i \overrightarrow{O_j M}}_{\text{vitesse d'entraînement}} + \underbrace{{}^iV_i^j(M)}_{\text{vitesse relative}} \qquad \text{(Eq.4.18)} $$

Si le point M est choisie à l'origine O_i de la famille $\mathfrak{f}_i$ (i.e. du repère $\mathcal{R}_i$), notons ${}^j\rho_i$ le vecteur $\overrightarrow{O_j O_i}$ pris dans le repère $\mathcal{R}_j$. Alors, l'algorithme permettant de déduire la vitesse linéaire au point O_i à partir de celle du point O_j s'écrit d'après la relation (Eq.4.18) et compte tenu de la bilinéarité du produit vectoriel et de l'orthogonalité de la matrice D_j^i par:

$$ {}^iv_i = D_j^i \, ({}^jv_j + {}^j\omega_j \times {}^j\rho_i) + {}^iv_i^j \qquad \text{(Eq.4.19)} $$

4.3.2.c Pseudo-accélération – Loi de composition des accélérations

Avant de définir ce que nous entendons par pseudo-accélération, rappelons tout d'abord que les accélérations angulaire et linéaire usuelles en mécanique classique ne définissent pas un champ équiprojectif, pour la simple raison que $\frac{d}{dt}[Y(M)] \neq \dot{Y}(M)$.

Définition 4.3

Nous appelons pseudo-accélération, la dérivée temporelle du champ des vitesses. On distingue:

- *La pseudo-accélération eulerienne qui découle de la dérivée du champ eulerien des vitesses, relativement à une famille fondamentale liée à l'espace, que l'on pourra noter* $\dot{U} = \frac{d}{dt}U$.
- *La pseudo-accélération lagrangienne induite par la dérivée du champ lagrangien des vitesses, relativement à une famille fondamentale liée au corps, que l'on pourra noter* $\dot{V} = \frac{d}{dt}V$.

Nous avons choisi d'appeler, une telle dérivée, pseudo-accélération pour deux raisons conflictuelles:

i) Sa dimension est celle d'une accélération, puisque elle s'obtient à partir de la dérivée par rapport au temps du torseur vitesse et donc elle s'apparente aux accélérations.

ii) Elle n'est pas exactement une une accélération, pour la simple raison que sa partie linéaire (moment en un point) n'en est pas une au sens physique du terme.

Ainsi, il nous semble tout à fait opportun de remarquer au lecteur que:

(Rm_1) Le vecteur résultant de la pseudo-accélération définit exactement l'accélération angulaire du mouvement.

(Rm$_2$) Quant au moment de la pseudo-accélération (i.e. pseudo-accélération linéaire), elle ne correspond pas exactement à l'accélération linéaire telle qu'elle est définie généralement en mécanique classique du point matériel.

(Rm$_3$) Toutefois, si besoin est, l'accélération linéaire γ_M du point mobile M peut être déduite du moment du torseur pseudo-accélération au point M, en lui rajoutant le terme $\omega_V \times V(M)$. i.e.:

$$\gamma_M = \dot{V}(M) + \omega_V \times V(M) \qquad \text{(Eq.4.20)}$$

(Rm$_4$) En conséquence, le calcul des accélération des particules d'un solide rigide, à un instant donné, se ramène à celui des torseurs V et $\dot{V}$ à cet instant.

Théorème et définition 4.6

La composition des pseudo-accélérations dans une paire cinématique $\mathbb{S}_{ij}$ se traduit par la relation:

$$^i\dot{V}_i = \boldsymbol{D}^i_{j*}(t)\, {}^j\dot{V}_j + \left[{}^iV_i, {}^iV^i_j\right] + {}^i\dot{V}^j_i \qquad \text{(Eq.4.21)}$$

Preuve :

La composition des pseudo-accélérations dans une paire cinématique $\mathbb{S}_{ij}$ se traduit par dérivation des membres de la relation (Eq.4.16). Ainsi, compte tenu du lemme 4.3:

$$^i\dot{V}_i = \boldsymbol{D}^i_{j*}\, {}^j\dot{V}_j + \left[\boldsymbol{D}^i_{j*}\, {}^jV_j, {}^iV^i_j\right] + {}^i\dot{V}^j_i \qquad \text{(Eq.4.22)}$$

or, d'après la relation (Eq.4.16), on peut écrire: $\boldsymbol{D}^i_{j*}\, {}^jV_j = {}^iV_i - {}^iV^j_i$.
de plus, nous savons que pour tout torseur X: $[X, X] = 0$.
Ainsi, de (Eq.4.22), il vient: $^i\dot{V}_i = \boldsymbol{D}^i_{j*}\, {}^j\dot{V}_j + \left[{}^iV_i, {}^iV^j_i\right] + {}^i\dot{V}^j_i$. ∎

Pour le modèle de calcul des pseudo-accélérations, nous préférons la relation (Eq.4.21) à (Eq.4.22) pour des raisons d'optimisation de la complexité algorithmique (i.e. nombre des opérations élémentaires en calcul de l'algorithme). En effet, lorsque nous écrirons, plus tard, l'algorithme complet qui permet de calculer les coordonnées des pseudo-accélérations, le calcul des coordonnées des vitesses aura été effectué à une étape antérieure. Ainsi, la relation (Eq.4.21) permet d'éviter des calculs redondants.

La traduction en vecteurs sommes de la loi de composition des pseudo-accélérations permet de déduire l'algorithme de composition des accélérations angulaires dans la paire cinématique $\mathbb{S}_{ij}$:

$$^i\dot{\omega}_i = D^i_j\, {}^j\dot{\omega}_j + {}^i\omega_i \times {}^i\omega^j_i + {}^i\dot{\omega}^j_i \qquad \text{(Eq.4.23)}$$

et l'évaluation de la relation (Eq.4.21) en un point M permet d'écrire:

$$\begin{aligned} ^i\dot{V}_i(M) = {} & D^i_j\, {}^j\dot{V}_j(O_j) + D^i_j\, {}^j\dot{\omega}_j \times D^i_j \overrightarrow{O_j M} + {}^iV_i(M) \times {}^i\omega^j_i + \\ & {}^i\omega_i \times {}^iV^j_i(M) + {}^i\dot{V}^j_i(M) \end{aligned} \qquad \text{(Eq.4.24)}$$

ainsi, lorsque le point M est pris à l'origine O_j, on obtient la relation permettant de calculé la pseudo-accélération linéaire du corps $\mathcal{C}_i$ au point O_i à partir de celle du corps $\mathcal{C}_j$ au point O_j. Ainsi, si ${}^j\rho_i = \overrightarrow{O_j O_i}$ est décrit par rapport au repère $\mathcal{R}_j$, alors:

$$ {}^i\dot{v}_i = D_j^i\,({}^j\dot{v}_j + {}^j\dot{\omega}_j \times {}^j\rho_i) + {}^iv_i \times {}^i\omega_i^j + {}^i\omega_i \times {}^iv_i^j + {}^i\dot{v}_i^j \qquad \text{(Eq.4.25)} $$

A ce stade du développement et compte tenu des remarques (Rm_1) et (Rm_3), il nous semble important de faire un certain nombre d'observations qui permettent de faire le lien avec la mécanique classique du point matériel:

(Ob_1) Le point matériel (abstraction mathématique du solide infiniment petit) se conçoit comme un cas particulier du solide rigide.

(Ob_2) Compte tenu de la remarque (Rm_3), la loi usuelle de composition des accélérations linéaires, en mécanique classique du point matériel, se déduit de la loi énoncée par la relation (Eq.4.24). En effet compte tenu des relations (Eq.4.17), (Eq.4.24) et (Eq.4.20) l'accélération absolue du point mobile M s'écrit:

$$ \begin{aligned} {}^i\gamma_M = {} & D_j^i\,{}^j\dot{V}_j(O_j) + D_j^i\,{}^j\dot{\omega}_j \times D_j^i\,\overrightarrow{O_jM} + D_j^i\,{}^j\omega_j \times {}^iV_i(M) + \\ & {}^i\omega_i \times {}^iV_i^j(M) + {}^i\dot{V}_i^j(M) \end{aligned} \qquad \text{(Eq.4.26)} $$

soit en remplaçant ${}^iV_i(M)$ par son expression dans la relation (Eq.4.18) et en prenant compte de la bilinéarité du produit vectoriel et de l'orthogonalité de l'opérateur D_j^i:

$$ \begin{aligned} {}^i\gamma_M = {} & D_j^i\left({}^j\dot{V}_j(O_j) + {}^j\omega_j \times {}^jV_j(O_j) + {}^j\omega_j \times {}^j\omega_j \times \overrightarrow{O_jM} + {}^j\dot{\omega}_j \times \overrightarrow{O_jM}\right) + \\ & 2\,D_j^i\,{}^j\omega_j \times {}^iV_i^j(M) + \\ & {}^i\omega_i^j \times {}^iV_i^j(M) + {}^i\dot{V}_i^j(M) \end{aligned} \qquad \text{(Eq.4.27)} $$

or ${}^j\dot{V}_j(O_j) + {}^j\omega_j \times {}^jV_j(O_j) = {}^j\gamma_{O_j}$ et ${}^i\omega_i^j \times {}^iV_i^j(M) + {}^i\dot{V}_i^j(M) = {}^i\gamma_{iM}^j$.
Donc, finalement:

$$ \begin{aligned} \underbrace{{}^i\gamma_M}_{\text{accélération absolue}} = {} & \underbrace{D_j^i\left({}^j\gamma_{O_j} + {}^j\omega_j \times {}^j\omega_j \times \overrightarrow{O_jM} + {}^j\dot{\omega}_j \times \overrightarrow{O_jM}\right)}_{\text{accélération d'entraînement}} + \\ & \underbrace{2\,D_j^i\,{}^j\omega_j \times {}^iV_i^j(M)}_{\text{accélération de Coriolis}} + \\ & \underbrace{{}^i\gamma_{iM}^j}_{\text{accélération relative}} \end{aligned} \qquad \text{(Eq.4.28)} $$

La relation (Eq.4.28) traduit la loi de composition des accélérations linéaires d'un point mobile M.

4.4 Cinétique du corps solide

Cette section s'inspire de l'exposé de D. Chevallier [CHEV-86] concernant la dynamique des solides rigides. Partant de l'idée de non-nécessité de décomposition systématique des articulations "complexes" (i.e. à plus d'un degré de liberté[1]) des systèmes articulés en paires prismatiques et pivots, il montre l'existence d'un opérateur généralisé d'inertie qu'il caractérise par un nombre de propriétés dans l'algèbre de Lie $\mathfrak{d}$. Dans le but d'appliquer ces résultats au calcul formel du modèle dynamique des systèmes articulés et en utilisant les outils de modélisation basés sur les nombres duaux, nous allons établir une forme pratique de cet opérateur généralisé d'inertie [A. Hamlili [HAML-91]].

4.4.1 Description de la masse inerte

La description spatiale de la répartition de masse dans un solide rigide est étroitement liée à la notion de mesure positive. Ainsi, si μ_s est la distribution de masse dans le solide rigide $\mathcal{C}$ dans une configuration $\boldsymbol{s}$, la masse m du solide est définie par:

$$m = \int_{\mathcal{C}} \mu_s(M) \qquad \text{(Eq.4.29)}$$

où M est un point matériel, du solide $\mathcal{C}$ dans la configuration $\boldsymbol{s}$, de masse infinitésimal $\mu_s(M)$. La masse totale doit nécessairement être indépendante du choix de la configuration $\boldsymbol{s}$ du solide $\mathcal{C}$. En effet, cette indépendance reflète le principe de l'invariance de la masse en mécanique newtonienne. Pour cela, il faut que la mesure positive μ satisfasse le relation:

$$\mu_{\boldsymbol{D}\bullet\boldsymbol{s}} = \boldsymbol{D}(\mu_s)\,, \quad \forall \boldsymbol{D} \in \mathbb{D} \qquad \text{(Eq.4.30)}$$

où $\boldsymbol{D}(\mu_s)$ est la mesure image de μ_s par l'application affine $\boldsymbol{D}$ (tout déplacement est une application affine de l'espace géométrique $\mathcal{E}$) définie par:

$$\boldsymbol{D}(\mu_s) = \mu_s(\boldsymbol{D}^{-1}(\mathcal{V}))\,; \quad \forall \mathcal{V} \in \mathcal{B}_D \qquad \text{(Eq.4.31)}$$

sachant que $\mathcal{B}_D$ est la tribu des ensembles $\mathcal{U}$ tels que: $\boldsymbol{D}^{-1}(\mathcal{U}) \in \mathcal{P}(\mathcal{E})$ (ensemble des parties de $\mathcal{E}$).

4.4.2 Description du centre de masse

En mécanique, on appelle centre de masse (ou centre d'inertie) du solide $\mathcal{C}$ à la configuration $\boldsymbol{s}$, le point G_s caractérisé par:

$$m\,\overrightarrow{OG_s} = \int_{\mathcal{C}} \overrightarrow{OM}\,\mu_s(M)\,; \quad \forall O \in \mathcal{E} \qquad \text{(Eq.4.32)}$$

Pour alléger les notations des calculs avenir, nous allons noter l'opérateur linéaire:

$$\mathbb{K}_s(O) = m\,\widetilde{\overrightarrow{OG_s}} \qquad \text{(Eq.4.33)}$$

tel que pour tout vecteur $\vec{u}$ de E, $\mathbb{K}_s(O)\,\vec{u} = m\,\overrightarrow{OG_s} \times \vec{u}$.

[1] Exemples: les poignets, les rotules, les articulations cylindriques... etc.

Propriétés 4.7

- *Soit D un élément du groupe $\mathbb{D}$ et D sa partie linéaire. Alors:*

$$\mathbb{K}_{D\bullet r} = D \circ \mathbb{K}_r \circ D^{-1} \qquad \text{(Eq.4.34)}$$

- *Soient M et N deux points de $\mathcal{E}$. Alors:*

$$\mathbb{K}_s(M) = \mathbb{K}_s(N) + m\,\widetilde{\overrightarrow{NM}} \qquad \text{(Eq.4.35)}$$

Preuve :

La démonstration est évidente. La relation (Eq.4.34) découle directement de (Eq.2.29) [Chapitre 3, page 51]. En suite, la relation (Eq.4.35) découle de la définition du tenseur $\mathbb{K}$. ∎

4.4.3 Tenseur d'inertie d'ordre deux

De façon générale, on définit le tenseur d'inertie du second ordre (ou tout simplement le tenseur d'inertie) en un point O_s lié au solide par:

$$\mathbb{I}_s(O_s)\,u = \int_C \overrightarrow{O_sM} \times (u \times \overrightarrow{O_sM})\, d\mu_s(M); \quad \forall u \in \mathbb{E} \qquad \text{(Eq.4.36)}$$

i.e.

$$\mathbb{I}_s(O_s) = -\int_C \widetilde{\overrightarrow{O_sM}}^2\, d\mu_s(M) \qquad \text{(Eq.4.37)}$$

Si (x, y, z) est le système de coordonnées cartésiennes associé à un repère (ou une famille fondamentale) d'origine le point O lié au solide, alors, la matrice associée à un tel opérateur s'écrit:

$$[\mathbb{I}_s(O_s)] = \begin{pmatrix} I_{x,x} & -I_{x,y} & -I_{x,z} \\ -I_{x,y} & I_{y,y} & -I_{y,z} \\ -I_{x,z} & -I_{y,z} & I_{z,z} \end{pmatrix} \qquad \text{(Eq.4.38)}$$

où les éléments scalaires de cette matrice sont donnés par les relations:

$$I_{x,x} = \int_C (y^2+z^2)\, d\mu_s(M), \quad I_{y,y} = \int_C (x^2+z^2)\, d\mu_s(M), \quad I_{z,z} = \int_C (x^2+y^2)\, d\mu_s(M),$$

$$I_{x,y} = \int_C x\,y\, d\mu_s(M), \qquad I_{x,z} = \int_C x\,z\, d\mu_s(M), \qquad I_{y,z} = \int_C y\,z\, d\mu_s(M).$$

avec $\mu_s(M)$ est la distribution de masse au point M du corps C qui occupe la configuration s. Lorsque le point O_s coïncide avec le centre d'inertie G_s, on obtient le tenseur central d'inertie.

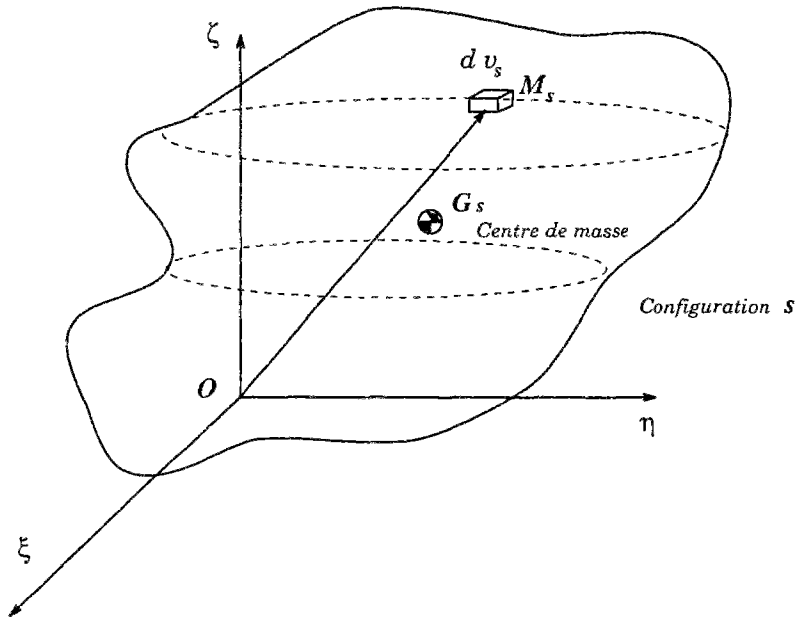
Théorème 4.8

Si $\mathbb{I}_s$ désigne Le tenseur d'inertie du second ordre lié au solide C, alors:

- *si O_s est un point lié au solide C, alors:*

$$\mathbb{I}_s(O_s) = \mathbb{I}_s(G_s) - m\,\widetilde{\rho_s^\star}^2 \qquad \text{(Eq.4.39)}$$

où $\rho_s^\star = \overrightarrow{O_sG_s}$. Cette relation traduit le théorème de Huygens.



La distribution de masse dans un solide peut être décrite à toute configuration s et tout instant. Si ϱ est la densité du corps C, la masse de l'objet infinitésimal M de volume dv_s est égale à $dm_s = d\mu_s(M) = \varrho\, dv_s$

Fig. 4.1: Distribution de masse dans un solide à une configuration donnée

- *si r est une configuration de $\mathcal{S}$, $\boldsymbol{D}$ un élément du groupe $\mathbb{D}$ et D la partie linéaire de $\boldsymbol{D}$, alors:*

$$\mathbb{I}_{D\bullet r} = \quad D \circ \mathbb{I}_r \circ D^{-1} \qquad \text{(Eq.4.40)}$$

Preuve :

La démonstration est évidente. Comme dans le cas précédent, la relation (Eq.4.39) découle directement de la définition du tenseur d'inertie d'ordre deux et la relation (Eq.4.40) découle de (Eq.2.29) [Chapitre 3, page 51]. ∎

4.4.4 Torseur cinétique – Opérateur généralisé d'inertie

Généralement, on définit le torseur cinétique comme étant le torseur dont le vecteur somme est donné par le vecteur quantité de mouvement et dont le moment est le moment cinétique. Le

but de ce paragraphe est de définir, si possible, un opérateur de $\mathfrak{V}$ qui permet d'obtenir le torseur cinétique à partir du torseur des vitesses. Ainsi, si nous arrivons à construire un tel opérateur, on aura démontré que la notion d'opérateur d'inertie se généralise à la description cinétique dans $\mathfrak{V}$ d'un mouvement quelconque.

D'une part, si nous notons $\overrightarrow{p}$ le vecteur quantité de mouvement:

$$\overrightarrow{p} = m\,V(G_s)$$

pour tout point O rigidement solidaire de $\mathcal{C}$, la vitesse linéaire du centre de gravité peut se réécrire de la forme:

$$V(G_s) = V(O) - \overrightarrow{OG_s} \times \omega_V$$

ainsi, le vecteur quantité de mouvement se réécrit en fonctions de l'opérateur $\mathbb{K}_s(O)$:

$$\overrightarrow{p} = m\,V(O) - \mathbb{K}_s(O)\,\omega_V \qquad \text{(Eq.4.41)}$$

D'autre part, le moment cinétique $P(O)$ au point O est donné par la relation:

$$P(O) = \int_{\mathcal{C}} \overrightarrow{OM} \times V(M)\, d\mu_s(M)$$

or $V(M) = V(O) + \omega_V \times \overrightarrow{OM}$, donc:

$$\begin{aligned} P(O) &= \left(\int_{\mathcal{C}} \widetilde{\overrightarrow{OM}}\, d\mu_s(M)\right) V(O) - \left(\int_{\mathcal{C}} \widetilde{\overrightarrow{OM}}^2\, d\mu_s(M)\right) \omega_V \\ &= \mathbb{K}_s(O)\,V(O) + \mathbb{I}_s(O)\,\omega_V \end{aligned} \qquad \text{(Eq.4.42)}$$

les relations (Eq.4.41) et (Eq.4.42) permettent de définir l'opérateur généralisé d'inertie en tout point O de $\mathcal{E}$ par:

$$\mathbb{H}_s(O) = \left[(-\mathbb{K}_s(O) + \varepsilon\,\mathbb{I}_s(O)) \circ Re + (\mathbb{M} + \varepsilon\,\mathbb{K}_s(O)) \circ Du\right] \circ \mathfrak{r}_O \qquad \text{(Eq.4.43)}$$

tel que si P désigne le torseur cinétique, alors: $P_O = \mathbb{H}_s(O)\,V$.
De plus, comme la détermination d'un torseur à partir de ses éléments de réduction en un point donné est unique, alors:

$$\mathbb{H}_s = \mathfrak{r}_O^{-1} \circ \left[(-\mathbb{K}_s(O) + \varepsilon\,\mathbb{I}_s(O)) \circ Re + (\mathbb{M} + \varepsilon\,\mathbb{K}_s(O)) \circ Du\right] \circ \mathfrak{r}_O \qquad \text{(Eq.4.44)}$$

est indépendant du choix du point O et on peut écrire:

$$P = \mathbb{H}_s\,V \qquad \text{(Eq.4.45)}$$

Proposition 4.9

Soient $\boldsymbol{r}$, $\boldsymbol{s}$ *deux configurations d'un corps* $\mathcal{C}$ *et* $\boldsymbol{D}$ *un déplacement du groupe* $\mathbb{D}$ *tels que* $\boldsymbol{s} = \boldsymbol{D} \bullet \boldsymbol{r}$:

(a) si $\mathfrak{c}_s \oplus \mathfrak{t} = \mathfrak{d}$ *est la décomposition en somme directe de l'algèbre de Lie* $\mathfrak{d}$ *au point* G_s *(centre d'inertie de* C *à la configuration* s*), alors:*

$$\mathbb{H}_s(\mathfrak{c}_s) \subseteq \mathfrak{t} \text{ et } \mathbb{H}_s(\mathfrak{t}) \subseteq \mathfrak{c}_s \qquad \text{(Eq.4.46)}$$

(b) les opérateurs $\mathbb{H}_r$ *et* $\mathbb{H}_s$ *relatifs aux configurations* r *et* s *du corps* C *sont liés par la relation:*

$$\mathbb{H}_s = \boldsymbol{D}_* \circ \mathbb{H}_r \circ \boldsymbol{D}_*^{-1} \qquad \text{(Eq.4.47)}$$

Preuve :

La démonstration est simple. Pour prouver la proposition (a), il suffit de voir que pour tout torseur X de $\mathfrak{d}$:

$$Re[\mathbb{H}_s(G_s)\,X] = m\,X(G_s) \text{ et } Du[\mathbb{H}_s(G_s)\,X] = \mathbb{I}_s(G_s)\,\omega_X$$

Quant à la démonstration de (b), il suffit de voir que $\mathbb{H}_s$ est $\boldsymbol{R}$–linéaire et donc elle découle (Eq.2.29) [Chapitre 3, page 51]. ∎

La relation mathématique (Eq.4.46) traduit le fait que le torseur cinétique engendré par le mouvement de rotation autour du centre d'inertie est un couple et le torseur cinétique engendré par une translation du solide est un glisseur dont l'axe passe par le centre d'inertie.

4.5 Torseur dynamique – Loi de Newton-Euler

Le torseur dynamique se définit comme étant la dérivée par rapport au temps du torseur cinétique. Pour établir une telle dérivée montrons le résultat préliminaire suivant:

Théorème 4.10

La dérivée de l'opérateur généralisé d'inertie est donnée par:

$$\frac{d}{dt}(\mathbb{H}_s) = ad\,V \circ \mathbb{H}_s - \mathbb{H}_s \circ ad\,V \qquad \text{(Eq.4.48)}$$

Preuve :

Soit une situation de référence fixe r. Notons $\boldsymbol{D}(t)$ le déplacement qui définit la trajectoire $s(t)$ par rapport à la configuration r. Alors, d'après la proposition 4.9 (b):

$$\begin{aligned} \frac{d}{dt}(\mathbb{H}_s) &= \frac{d}{dt}(\boldsymbol{D}_* \circ \mathbb{H}_r \circ \boldsymbol{D}_*^{-1}) \\ &= \frac{d}{dt}\boldsymbol{D}_*(t) \circ \mathbb{H}_r \circ \boldsymbol{D}_*^{-1} + \boldsymbol{D}_*(t) \circ \mathbb{H}_r \circ \frac{d}{dt}\boldsymbol{D}_*^{-1} \end{aligned} \qquad \text{(Eq.4.49)}$$

Ce qui montre la relation (Eq.4.48) compte tenu du théorème 4.4. ∎

Corollaire 4.11

Le torseur dynamique est donné par:

$$\frac{d}{dt}P_s = \mathbb{H}_s(\dot{V}) + [V, \mathbb{H}_s(V)] \qquad \text{(Eq.4.50)}$$

Preuve :

Il suffit de voir que $\mathbb{H}_s$ est un opérateur $\boldsymbol{R}$-linéaire sur $\mathfrak{V}$, donc:

$$\frac{d}{dt}\left(\mathbb{H}_s(V)\right) = \mathbb{H}_s(V) + \mathbb{H}_s(\dot{V}) \qquad \text{(Eq.4.51)}$$

et comme $[V, V] = 0$, compte tenu de (Eq.4.48), la relation (Eq.4.50) découle directement de la relation (Eq.4.51). ∎

Connaissant les caractéristiques inertielles d'un solide, les lois d'Euler et de Newton permettent de déterminer, par le calcul, les efforts (i.e. forces et couples) inertiels qui agissent en son centre de masse. La représentation torsoriel des forces et couples permet d'exprimer ces deux lois sous une forme synthétique en terme du torseur dynamique. Rappelons ici que nous admettons l'objectivité des efforts inertiels[2]. Ainsi, Soit $\mathcal{F}^e$ le torseur résultant des efforts extérieurs agissant sur le solide $\mathcal{C}$, alors:

$$\mathcal{F}_s^e = \frac{d}{dt} P_s \qquad \text{(Eq.4.52)}$$

cette loi sera dite loi fondamentale de Newton-Euler et nous l'énoncerons souvent, compte tenu de la relation (Eq.4.50), sous la forme:

$$\mathcal{F}_s^e = \mathbb{H}_s(\dot{V}) + [V, \mathbb{H}_s(V)] \qquad \text{(Eq.4.53)}$$

Explicitons cette relation pour voir comment s'identifient et s'interprètent les grandeurs vectorielles qui découlent de sa réduction en point. Revenons à la relation (Eq.4.50):

$$\frac{d}{dt} P_s = \mathbb{H}_s(\dot{V}) + [V, \mathbb{H}_s(V)]$$

en tout point O de $\mathcal{E}$ en notant $\rho^\star = \overrightarrow{OG}$, la partie réelle de $\left[\frac{d}{dt} P_s\right]_O = \tau_O\left(\frac{d}{dt} P_s\right)$ est donnée par:

$$\begin{aligned} Re\left[\frac{d}{dt} P_s\right]_O &= m\left(\dot{V}(O) - \rho_s^\star \times \dot{\omega}_V + \omega_V \times \left(V(O) - \rho_s^\star \times \omega_V\right)\right) \\ &= m\underbrace{\left(\underbrace{\left(\dot{V}(O) + \omega_V \times V(O)\right)}_{\text{accélération du corps } \mathcal{C} \text{ au point } O} + \dot{\omega}_V \times \rho_s^\star + \omega_V \times \left(\omega_V \times \rho_s^\star\right)\right)}_{\text{accélération au centre de masse } G_s \text{ du corps } \mathcal{C}} \end{aligned} \qquad \text{(Eq.4.54)}$$

ce qui donne exactement la force d'inertie au centre de masse G_s (loi fondamentale de la dynamique).

[2]Les lecteurs soucieux de cette question peuvent se rapporter aux références indiqués dans notre bibliographie et notamment à [H. Poincaré [POIN-02], J. Ullmo [ULLM-65] pour les aspects philosophiques de la question, P. Germain [GERM-86] pour l'aspect pratique].

Maintenant, calculons le moment N^e, au point O, de la dérivée temporelle du torseur cinétique $P(t)$. Il est donné par la partie duale du vecteur dual $\left[\frac{d}{dt} P_s\right]_O$. Donc:

$$Du\left[\frac{d}{dt} P_s\right]_O = \underbrace{\mathbb{I}_s(O)\dot{\omega}_V + \omega_V \times \mathbb{I}_s(O)\omega_V}_{\text{moment cinétique dû à la rotation autour du point } O} + \underbrace{\underbrace{m\left(\dot{V}(O) + \omega_V \times V(O)\right)}_{\text{accélération du corps } C \text{ au point } O} \times \rho_s^\star}_{\text{moment cinétique dû au mouvement linéaire du point } O} \quad \text{(Eq.4.55)}$$

Cela donne, exactement, le moment résultant au point O. En effet, la relation qui exprime l'accélération du point O en fonction de celle du centre de masse G n'est autre que:

$$\dot{V}(O) + \omega_V \times V(O) = \dot{V}(G_s) + \omega_V \times V(G_s) - \dot{\omega}_V \times \rho_s^\star + - \omega_V \times (\omega_V \times \rho_s^\star) \quad \text{(Eq.4.56)}$$

en rapportant la relation (Eq.4.56) dans (Eq.4.55) et compte tenu du théorème de Huygens, on obtient donc:

$$Du\left[\frac{d}{dt} P_s\right]_O = \underbrace{\mathbb{I}_s(G_s)\dot{\omega}_V + \omega_V \times \mathbb{I}_s(G_s)\omega_V}_{\text{moment résultant dû au mouvement propre de } C} + \underbrace{\underbrace{m\left(\dot{V}(G_s) + \omega_V \times V(G_s)\right)}_{\text{accélération du corps } C \text{ au point } G_s} \times \rho_s^\star}_{\text{moment de la force d'inertie au point } O} \quad \text{(Eq.4.57)}$$

or, comme nous savons que: $\mathcal{F}_s^e(G_s) = \mathbb{I}_s(G_s)\dot{\omega}_V + \omega_V \times \mathbb{I}_s(G_s)\omega_V$

et $\omega_{\mathcal{F}_s^e} = m\left(\dot{V}(G_s) + \omega_V \times V(G_s)\right)$. Donc, on a:

$$\mathcal{F}_s^e(O) = N^e = Du\left[\frac{d}{dt} P_s\right]_O \quad \text{(Eq.4.58)}$$

le résultat exprimé par la relation (Eq.4.58) est connu sous le nom du théorème du moment cinétique.

4.6 Modèle dynamique des robots à chaîne ouverte simple

On considère dans ce paragraphe simplement les robots manipulateurs à un seul organe terminal et des mécanismes à chaîne ouverte simple. Le modèle dynamique des robots à plusieurs organes terminaux et mécanismes à structure arborescente feront l'objet d'une section à part.

Définition 4.4

Toute loi qui présente le calcul d'une quantité indicée sur un ensemble totalement ordonné à partir de sa valeur au rang précédant ou succédant est dite récurrente.

(a) Si ce calcul fait appel au rang précédent, la récurrence est dite directe (i.e. ascendante).

(b) Si, au contraire, il fait appel au rang succédant, la récurrence est dite inverse (i.e. descendante).

Nous allons considérer maintenant le problème du calcul des efforts généralisés $\underline{Q}$ qui correspondent à une trajectoire donnée du système articulé, chaque élément du robot ou du système articulé étant supposé un corps rigide. Aussi, nous supposerons connues, les positions, vitesses et accélérations généralisées $(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}})$ des éléments du robot qui caractérisent, ainsi, complètement sa cinématique. Les opérateurs généralisés d'inertie $\mathbb{H}_i$ (relatifs aux corps C_i et pris aux origines O_i des familles fondamentales $\mathfrak{f}_i$ rigidement solidaires aux corps C_i) seront également supposés connus pour chaque élément C_i du système articulé. Dans le calcul du modèle dynamique, on s'intéresse à décrire toutes les quantités mécaniques relativement à un référentiel de laboratoire (ou de travail, souvent lié au socle, quand il s'agit de robots manipulateurs). Ainsi, nous allons privilégier la description lagrangienne du mouvement. S'il y a besoin d'utiliser la représentation eulerienne du mouvement, les relations correspondantes pourront être retrouvées comme nous l'avons décrit au paragraphe 4.2.2. De même, dans notre analyse, nous allons privilégier des articulations cylindriques, les articulations classiques de la robotiques (mouvement à un degré de liberté) peuvent être retrouvées en annulant les paramètres inutiles (i.e. cas des articulations pivots et prismatiques) ou en imposant une relation de dépendance entre ces paramètres (i.e. cas des articulations hélicoïdales).

4.6.1 Algorithmes récurrents pour le calcul des vitesses et accélérations

On conserve ici les mêmes conventions d'indiçage -des éléments d'un système articulé et de ses articulations- définis au paragraphe 3.9.1 [Chapitre 3, page 88]. A chaque articulation, nous allons associer une famille $\mathfrak{f}_i = (\xi_i, \eta_i, \zeta_i)$ telle que le glisseur ζ_i définisse l'axe de l'articulation d'indice i. Ainsi, nous proposerons pour le calcul des vitesses et accélérations une méthode récurrente directe, partant du corps d'indice 0 et s'arrêtant au corps d'indice N (i.e. organe terminal).

4.6.1.a "Propagation" des champs cinématiques

D'une part, d'après la relation (Eq.4.16), la propagation[3] des vitesses est donnée par:

$$ {}^{i+1}V_{i+1} = \boldsymbol{D}_{i*}^{i+1}\, {}^{i}V_i + {}^{i+1}V_{i+1}^{i} \qquad \text{(Eq.4.59)} $$

or la paire $\mathbb{S}_{i+1} = \{C_i, C_{i+1}\}$ étant une paire cylindrique, notons $\alpha_{i+1} = \theta_{i+1} + \varepsilon\, d_{i+1}$ le paramètre dual associé aux deux degrés de liberté d'une telle paire (où θ_{i+1} est le degré de liberté de rotation autour de l'axe $\Lambda_{\zeta_{i+1}}$ de la paire $\mathbb{S}_{i+1}$ et d_{i+1} le degré de liberté de translation le long de l'axe $\Lambda_{\zeta_{i+1}}$). Alors, d'après la relation (Eq.4.6), on a:

$$ {}^{i+1}V_{i+1}^{i} = \dot{\alpha}_{i+1}\, {}^{i+1}\zeta_{i+1} \qquad \text{(Eq.4.60)} $$

[3] Il faudrait comprendre ici le mot "propagation" dans le sens figuré relatif au mode du calcul récurrent (les termes de rang supérieurs sont exprimés en fonction de leurs prédécesseurs).

Ainsi, l'équation (Eq.4.59) se met sous la forme:

$$^{i+1}V_{i+1} = \boldsymbol{D}_{i*}^{i+1}\, {}^{i+1}V_i + \dot{\alpha}_{i+1}\, {}^{i+1}\zeta_{i+1} \qquad \text{(Eq.4.61)}$$

d'après la relation (Eq.4.21), la "propagation" des pseudo-accélérations est obtenue par la relation:

$$^{i+1}\dot{V}_{i+1} = \boldsymbol{D}_{i*}^{i+1}\, {}^{i}\dot{V}_i + \left[\boldsymbol{D}_{i*}^{i+1}\, {}^{i}V_i,\, {}^{i+1}V_{i+1}^{i}\right] + {}^{i+1}\dot{V}_{i+1}^{i} \qquad \text{(Eq.4.62)}$$

et en dérivant par rapport au temps la relation (Eq.4.60):

$$^{i+1}\dot{V}_{i+1}^{i} = \ddot{\alpha}_{i+1}\, {}^{i+1}\zeta_{i+1} \qquad \text{(Eq.4.63)}$$

la relation (Eq.4.62) devient:

$$^{i+1}\dot{V}_{i+1} = \boldsymbol{D}_{i*}^{i+1}\, {}^{i}\dot{V}_i + \dot{\alpha}_{i+1}\left[\boldsymbol{D}_{i*}^{i+1}\, {}^{i}V_i,\, {}^{i+1}\zeta_{i+1}\right] + \ddot{\alpha}_{i+1}\, {}^{i+1}\zeta_{i+1} \qquad \text{(Eq.4.64)}$$

d'où l'algorithme suivant:

Algorithme 4.1

- **Cas général:**

$$^{i+1}V_{i+1} = \boldsymbol{D}_{i*}^{i+1}\, {}^{i}V_i + {}^{i+1}V_{i+1}^{i}$$

$$^{i+1}\dot{V}_{i+1} = \boldsymbol{D}_{i*}^{i+1}\, {}^{i}\dot{V}_i + \left[\boldsymbol{D}_{i*}^{i+1}\, {}^{i}V_i,\, {}^{i+1}V_{i+1}^{i}\right] + {}^{i+1}\dot{V}_{i+1}^{i}$$

- **Cas des articulations de type C:**

$$^{i+1}V_{i+1} = \boldsymbol{D}_{i*}^{i+1}\, {}^{i+1}V_i + \dot{\alpha}_{i+1}\, {}^{i+1}\zeta_{i+1}$$

$$^{i+1}\dot{V}_{i+1} = \boldsymbol{D}_{i*}^{i+1}\, {}^{i}\dot{V}_i + \dot{\alpha}_{i+1}\left[\boldsymbol{D}_{i*}^{i+1}\, {}^{i}V_i,\, {}^{i+1}\zeta_{i+1}\right] + \ddot{\alpha}_{i+1}\, {}^{i+1}\zeta_{i+1}$$

Sauf mention contraire, dans toute la suite, on ne considère que des articulations admettant un axe fixe -dans le temps- (i.e. des articulations de type R, P, H ou C).

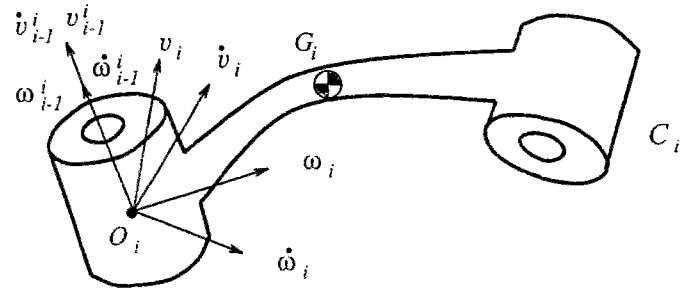
4.6.1.b "Propagation" des vitesses et pseudo-accélérations angulaires et linéaires

Notons O_i et O_{i+1} les origines respectifs des familles $\mathfrak{f}_i = (\xi_i, \eta_i, \zeta_i)$ et $\mathfrak{f}_{i+1} = (\xi_{i+1}, \eta_{i+1}, \zeta_{i+1})$ rigidement solidaires au corps $\mathcal{C}_i$ et $\mathcal{C}_{i+1}$ et telles que le glisseur ζ_{i+1} définit l'axe de l'articulation de la paire cylindrique $\mathbb{S}_{i+1} = \{\mathcal{C}_i, \mathcal{C}_{i+1}\}$. Ainsi, la partie réelle de la relation (Eq.4.59) permet d'exprimer la loi de propagation des vitesses angulaires:

$$^{i+1}\omega_{i+1} = D_i^{i+1}\, {}^{i}\omega_i + \dot{\theta}_{i+1}\, {}^{i+1}z_{i+1} \qquad \text{(Eq.4.65)}$$

et sa partie duale exprime la loi de propagation des vitesses linéaires aux origines des familles fondamentales solidaires aux éléments du robot, ainsi:

$$^{i+1}v_{i+1} = D_i^{i+1}\left({}^{i}v_i + {}^{i}\omega_i \times {}^{i}\rho_{i+1}\right) + \dot{d}_{i+1}\, {}^{i+1}z_{i+1} \qquad \text{(Eq.4.66)}$$



Les grandeurs cinématiques caractérisant le mouvement du corps $\mathcal{C}_i$ sont évalué au point O_i (Origine de la famille $\mathfrak{f}_i$) rigidement solidaire à $\mathcal{C}_i$.

Fig. 4.2: Schématisation des grandeurs cinématiques

où ${}^i\rho_{i+1}$ désigne le vecteur $\overrightarrow{O_i O_{i+1}}$ pris dans le repère $\mathcal{R}_i$ qui matérialise la famille $\mathfrak{f}_i$.

Maintenant, la partie réelle de l'équation (Eq.4.64) donne la loi de propagation des accélérations angulaires:

$$ {}^{i+1}\dot{\omega}_{i+1} = D_j^{i+1}\,{}^j\dot{\omega}_j + {}^{i+1}\omega_{i+1} \times \dot{\theta}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}z_{i+1} \qquad \text{(Eq.4.67)} $$

Quant à la loi de propagation des pseudo-accélérations linéaires aux origines des familles fondamentales $\mathfrak{f}_i$ et $\mathfrak{f}_{i+1}$, elle est obtenue par la partie duale de l'équation (Eq.4.64), soit:

$$ \begin{aligned} {}^{i+1}\dot{v}_{i+1} = {} & D_i^{i+1}\,({}^i\dot{v}_i + {}^i\dot{\omega}_i \times {}^i\rho_{i+1}) + {}^{i+1}v_{i+1} \times \dot{\theta}_{i+1}\,{}^{i+1}z_{i+1} + \\ & {}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{d}_{i+1}\,{}^{i+1}z_{i+1} \end{aligned} \qquad \text{(Eq.4.68)} $$

Algorithme 4.2

$$ {}^{i+1}\omega_{i+1} = D_i^{i+1}\,{}^i\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}z_{i+1} $$

$$ {}^{i+1}v_{i+1} = D_i^{i+1}\,({}^iv_i + {}^i\omega_i \times {}^i\rho_{i+1}) + \dot{d}_{i+1}\,{}^{i+1}z_{i+1} $$

$$ {}^{i+1}\dot{\omega}_{i+1} = D_j^{i+1}\,{}^j\dot{\omega}_j + {}^{i+1}\omega_{i+1} \times \dot{\theta}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}z_{i+1} $$

$$ \begin{aligned} {}^{i+1}\dot{v}_{i+1} = {} & D_i^{i+1}\,({}^i\dot{v}_i + {}^i\dot{\omega}_i \times {}^i\rho_{i+1}) + {}^{i+1}v_{i+1} \times \dot{\theta}_{i+1}\,{}^{i+1}z_{i+1} + \\ & {}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{d}_{i+1}\,{}^{i+1}z_{i+1} \end{aligned} $$

4.6.1.c Algorithmes dans le cas des articulations prismatiques et pivots

Les articulations pivots et prismatiques sont des articulations très utilisées dans la réalisation technologique des systèmes articulés en général et particulièrement en robotique.

- **Cas des articulations pivots:**
  Pour obtenir le modèle dans le cas des articulations de type pivot, il suffit d'annuler dans les équations de l'algorithme 4.2 les occurrences des paramètres d_{i+1}, $\dot{d}_{i+1}$ et $\ddot{d}_{i+1}$:

Algorithme 4.3

$$
\begin{aligned}
{}^{i+1}\omega_i &= D_i^{i+1}\,{}^{i}\omega_i \\
{}^{i+1}\omega_{i+1} &= {}^{i+1}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}z_{i+1} \\
{}^{i+1}v_{i+1} &= D_i^{i+1}\,{}^{i}v_i + {}^{i+1}\omega_{i+1} \times {}^{i+1}\rho_{i+1} \\
{}^{i+1}\dot{\omega}_i &= D_i^{i+1}\,{}^{i}\dot{\omega}_i \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}\dot{\omega}_i + {}^{i+1}\omega_{i+1} \times \dot{\theta}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}z_{i+1} \\
{}^{i+1}\dot{v}_{i+1} &= D_i^{i+1}\,{}^{i}\dot{v}_i + {}^{i+1}\dot{\omega}_i \times {}^{i+1}\rho_{i+1} + {}^{i+1}v_{i+1} \times \dot{\theta}_{i+1}\,{}^{i+1}z_{i+1}
\end{aligned}
$$

- **Cas des articulations prismatiques:**
  De façon analogue, pour obtenir le modèle dans le cas des articulations prismatiques, il suffit d'annuler dans les équations de l'algorithme 4.2 les paramètres angulaires θ_{i+1}, $\dot{\theta}_{i+1}$ et $\ddot{\theta}_{i+1}$:

Algorithme 4.4

$$
\begin{aligned}
{}^{i+1}\omega_{i+1} &= D_i^{i+1}\,{}^{i}\omega_i \\
{}^{i+1}v_{i+1} &= D_i^{i+1}\,{}^{i}v_i + {}^{i+1}\omega_{i+1} \times {}^{i+1}\rho_{i+1} + \dot{d}_{i+1}\,{}^{i+1}z_{i+1} \\
{}^{i+1}\dot{\omega}_{i+1} &= D_i^{i+1}\,{}^{i}\dot{\omega}_i \\
{}^{i+1}\dot{v}_{i+1} &= D_i^{i+1}\,{}^{i}\dot{v}_i + {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}\rho_{i+1} + {}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{d}_{i+1}\,{}^{i+1}z_{i+1}
\end{aligned}
$$

- **Implémentation itérative mixte:**
  Dans le cas de mécanismes qui ne comportent que des articulations prismatiques et pivots, afin de donner une formulation mixte qui traite aussi bien les articulations prismatiques et rotoïdes, nous allons utiliser les classiques[4]partitions binaires $(\sigma_i, \overline{\sigma}_i)$ de l'unité. Ainsi, pour tout entier $i \in [1..N]$ (où N est le nombre de degrés de liberté du robot ou de la chaîne

[4]Les partitions binaires de l'unité sont très utilisées en algorithmique pour désigner le processus conditionnel:

si $\cdots$ alors $\cdots$
sinon $\cdots$

cinématique ouverte simple), nous allons définir la partition $(\sigma_i, \overline{\sigma}_i)$ par:

$$\left.\begin{array}{lll} \circ\ \sigma_i = 1 & \text{si l'articulation } i \text{ est rotoïde} \\ \circ\ \sigma_i = 0 & \text{si l'articulation } i \text{ est prismatique} \\ \circ\ \overline{\sigma}_i = 1 - \sigma_i & \text{cœfficient conjugué de } \sigma_i \end{array}\right\} \Rightarrow \sigma_i + \overline{\sigma}_i = 1$$

les coordonnées articulaires θ_i et d_i peuvent donc se mettre sous forme d'une notation unifiée grâce au changement de variables:

$$q_i = \sigma_i\,\theta_i + \overline{\sigma}_i\,d_i \qquad \text{(Eq.4.69)}$$

Ainsi, les deux algorithmes 4.3 et 4.4 se réduisent à un unique algorithme incluant à la fois les deux types d'articulations prismatique et pivot:

Algorithme 4.5

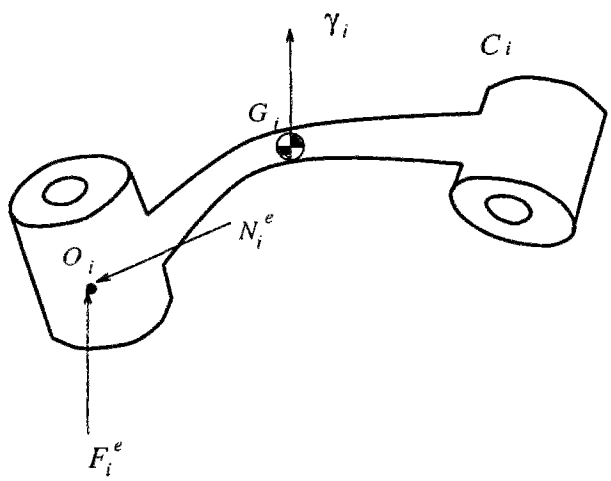
$$\begin{aligned}
{}^{i+1}\omega_i &= D_i^{i+1}\,{}^i\omega_i \\
{}^{i+1}\omega_{i+1} &= {}^{i+1}\omega_i + \sigma_{i+1}\,\dot{q}_{i+1}\,{}^{i+1}z_{i+1} \\
{}^{i+1}v_i &= D_i^{i+1}\,{}^i v_i \\
{}^{i+1}v_{i+1} &= {}^{i+1}v_i + {}^{i+1}\omega_i \times {}^{i+1}\rho_{i+1} + \overline{\sigma}_{i+1}\,\dot{q}_{i+1}\,{}^{i+1}z_{i+1} \\
{}^{i+1}\dot{\omega}_i &= D_i^{i+1}\,{}^i\dot{\omega}_i \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}\dot{\omega}_i + \sigma_{i+1}\left({}^{i+1}\omega_{i+1} \times \dot{q}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{q}_{i+1}\,{}^{i+1}z_{i+1}\right) \\
{}^{i+1}\dot{v}_i &= D_i^{i+1}\,{}^i\dot{v}_i \\
{}^{i+1}\dot{v}_{i+1} &= {}^{i+1}\dot{v}_i + {}^{i+1}\dot{\omega}_i \times {}^{i+1}\rho_{i+1} + \sigma_{i+1}\,{}^{i+1}v_{i+1} \times \dot{q}_{i+1}\,{}^{i+1}z_{i+1} + \\
&\qquad \overline{\sigma}_{i+1}\left({}^{i+1}\omega_{i+1} \times \dot{q}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{q}_{i+1}\,{}^{i+1}z_{i+1}\right)
\end{aligned}$$

4.6.2 Application des lois de Newton-Euler

Connaissant les familles fondamentales $\mathfrak{f}_i$ et les caractéristiques inertiels (i.e. les opérateurs d'inertie $\mathbb{H}_i$) associées aux éléments $\mathcal{C}_i$ d'un robot ou d'un système articulé à chaîne cinématique ouverte simple dans le cas général, il est possible de connaître les efforts qui agissent sur les corps $\mathcal{C}_i$. Il suffit pour cela d'exprimer la loi fondamentale de Newton-Euler (Eq.4.53) pour chaque corps $\mathcal{C}_i$ et d'évaluer les expressions obtenues respectivement aux points O_i associés:

$${}^i\mathcal{F}_i^e = {}^i\mathbb{H}_i({}^i\dot{V}_i) + [{}^iV_i, {}^i\mathbb{H}_i({}^iV_i)] \qquad \text{(Eq.4.70)}$$

la séparation des parties réelle et duale de la relation (Eq.4.70) permet d'obtenir les relations:



Les vecteurs ${}^iF_i^e$ est proportionnel à l'accélération ${}^i\gamma_i$ du centre de masse G_i. Ces deux vecteurs sont tels que ${}^iF_i^e = m_i\,{}^i\gamma_i$ [Relation (Eq.4.71)].

Fig. 4.3: Résultantes des efforts agissant sur un élément

- <u>La loi de Newton</u>:

$$ {}^iF_i^e = m_i\,({}^i\dot{v}_i + {}^i\dot{\omega}_i \times {}^i\rho_i^\star + {}^i\omega_i \times ({}^iv_i + {}^i\omega_i \times {}^i\rho_i^\star)) \qquad \text{(Eq.4.71)} $$

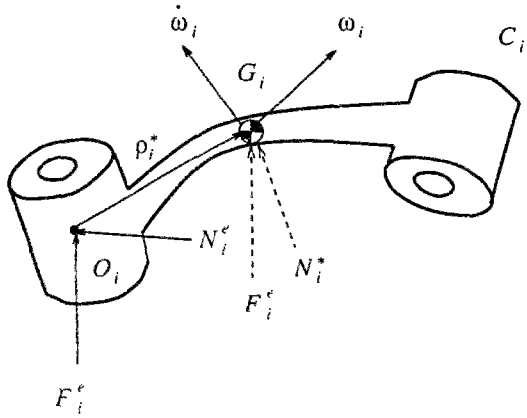
où m_i désigne la masse du corps $\mathcal{C}_i$.

- <u>Le théorème d'Euler au point O_i</u>: si ${}^i\mathbb{I}_i$ exprime le tenseur d'inertie au centre de masse G_i, alors, le théorème d'Euler au point O_i découle des relations (Eq.4.57) et (Eq.4.58):

$$ {}^iN_i^e = {}^i\mathbb{I}_i\,{}^i\dot{\omega}_i + {}^i\omega_i \times {}^i\mathbb{I}_i\,{}^i\omega_i + {}^i\rho_i^\star \times {}^iF_i^e \qquad \text{(Eq.4.72)} $$

4.6.3 Efforts des actionneurs au niveau des articulations

Il s'agit maintenant d'établir les loi de "propagation" des efforts moteurs qui produisent le mouvement de la structure articulée. Comme dans les paragraphes précédents, les calculs vont être effectués sur les grandeurs dynamiques avec la description lagrangienne des efforts. Notons, $\mathcal{T}_i^{i-1}$ le champ résultant des forces et couples qui agissent au niveau de l'actionneur de l'articulation d'indice i sur corps $\mathcal{C}_i$ par le corps $\mathcal{C}_{i-1}$. Dans de telles notations, d'après le principe des actions



Lorsqu'une force est considérée en un point autre que son point d'application elle induit un couple correctif égal au moment de la force en ce point. Ainsi, ${}^iN_i^e = {}^iN_i^* + {}^iF_i^e \times {}^i\rho_i^*$, où ${}^iN_i^* = {}^i\mathbb{I}_i{}^i\dot{\omega}_i + {}^i\omega_i \times {}^i\mathbb{I}_i{}^i\omega_i$.

Fig. 4.4: Correction associée au déplacement d'une force

mutuelles, le champ résultant des forces et moments agissant au niveau de l'articulation $i+1$ sur corps $\mathcal{C}_i$ par le corps $\mathcal{C}_{i+1}$, s'écrit relativement à la famille $\mathfrak{f}_i$ sous la forme:

$$ {}^i\mathcal{T}_i^{i+1} = -\boldsymbol{D}_{i+1_*}^i {}^{i+1}\mathcal{T}_{i+1}^i \qquad \text{(Eq.4.73)} $$

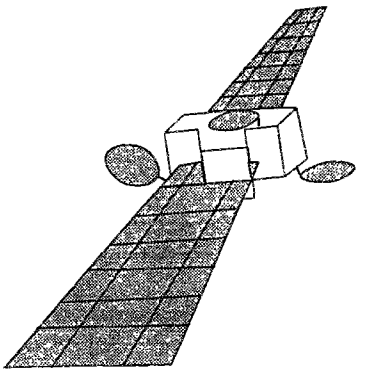
Dans le cas général, le bilan des forces et couples appliqués au corps $\mathcal{C}_i$ permet d'écrire:

$$ {}^i\mathcal{F}_i^e = {}^i\mathcal{T}_i^{i-1} + {}^i\mathcal{T}_i^{i+1} + {}^i\mathcal{H}_i \qquad \text{(Eq.4.74)} $$

où le torseur $\mathcal{H}_i$ désigne le champ résultant des forces et couples agissant sur le corps $\mathcal{C}_i$, mis à part les champs ${}^i\mathcal{T}_i^{i-1}$ et ${}^i\mathcal{T}_i^{i+1}$. Ce champ peut représenter toutes sortes d'actions appliquées au corps $\mathcal{C}_i$ (frottements, résistance de l'aire dans le cas de mouvements à grande vitesses, vents, gravitation, ...etc) [Fig. 4.5].

Pour fixer les idées, si on suppose que les seuls efforts exercés sur le corps $\mathcal{C}_i$, mis à part les champs ${}^i\mathcal{T}_i^{i-1}$ et ${}^i\mathcal{T}_i^{i+1}$, dérivent de l'action du champ de gravité terrestre. Ainsi, si $\vec{g}$ désigne le vecteur gravité, ${}^ig$ sa représentation relative au repère $\mathcal{R}_i$ -associé à la famille $\mathfrak{f}_i$- et si on note $\mathcal{G}_i$ le glisseur de vecteur $\vec{g}$ dont l'axe $\Lambda_{\mathcal{G}_i}$ passe par le centre de gravité G_i du corps $\mathcal{C}_i$, le bilan des forces et couples appliqués au corps $\mathcal{C}_i$ permet d'écrire tout simplement:

$$ {}^i\mathcal{F}_i^e = {}^i\mathcal{T}_i^{i-1} + {}^i\mathcal{T}_i^{i+1} + m_i\,{}^i\mathcal{G}_i $$



Un satellite placé en orbite est soumis à plusieurs forces, entre autres, la gravité terrestre, la gravité lunaire (ainsi que les gravités des autres planètes), les efforts exercés par les vents de radiations et de particules ... etc. Dans ce cas, pour chaque élément du satellite, le torseur $\mathcal{H}_i$ doit prendre en compte les effets de tous ces efforts.

Fig. 4.5: Satellite géostationnaire Anik E

soit en tenant compte de la relation (Eq.4.73), il vient:

$$ {}^i\mathcal{F}_i^e = {}^i\mathcal{T}_i^{i-1} - \boldsymbol{D}_{i+1*}^i \, {}^{i+1}\mathcal{T}_{i+1}^i + m_i \, {}^i\mathcal{G}_i $$

Ainsi, nous obtenons la relation récurrente (dans l'algèbre de Lie $\mathfrak{d}$) permettant de déduire le champ $\mathcal{T}_i^{i-1}$ du champ ${}^{i+1}\mathcal{T}_{i+1}^i$:

$$ {}^i\mathcal{T}_i^{i-1} = {}^i\mathcal{F}_i^e + \boldsymbol{D}_{i+1*}^i \, {}^{i+1}\mathcal{T}_{i+1}^i - m_i \, {}^i\mathcal{G}_i \qquad \text{(Eq.4.75)} $$

Pour déduire les expressions vectorielles des algorithmes récurrents calculant les forces et moments agissant aux niveau de l'articulation d'indice i à l'origine O_i de la famille fondamentale $\mathfrak{f}_i$ relative au corps $\mathcal{C}_i$, nous allons écrire:

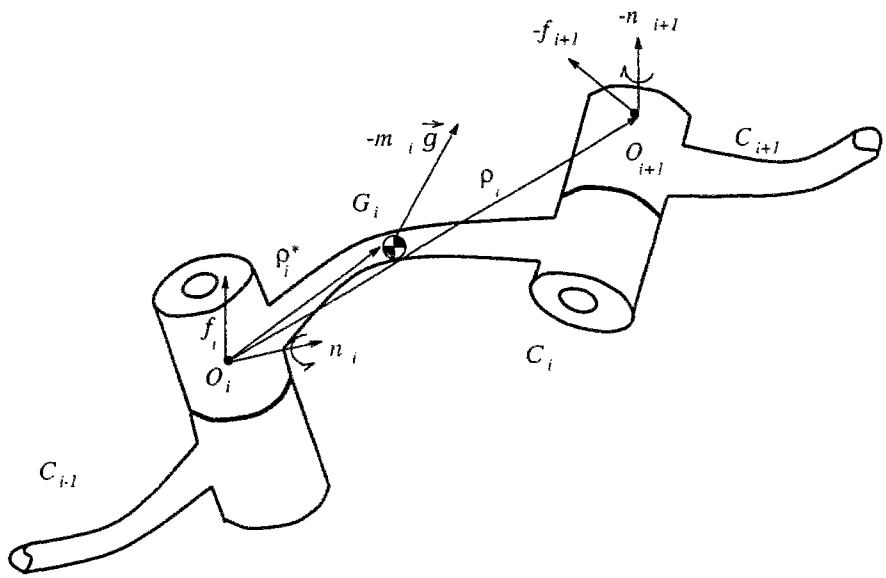
$$ {}^i\mathcal{T}_i^{i-1} = {}^i f_i + \varepsilon \, {}^i n_i \qquad \text{(Eq.4.76)} $$

où ${}^i f_i$ désigne la force exercée sur le corps $\mathcal{C}_i$ par le corps $\mathcal{C}_{i-1}$ et ${}^i n_i$ le couple exercé, au point O_i, sur le corps $\mathcal{C}_i$ par le corps $\mathcal{C}_{i-1}$. La séparation des parties réelle et duale de la relation (Eq.4.75), compte tenue de l'égalité (Eq.4.76), permet d'écrire, d'une part:

$$ {}^i f_i = {}^i F_i^e + D_{i+1}^i \, {}^{i+1} f_{i+1} - m_i \, {}^i g \qquad \text{(Eq.4.77)} $$

et, d'autre part:

$$ {}^i n_i = {}^i N_i^e + D_{i+1}^i \left({}^{i+1} n_{i+1} + {}^{i+1} f_{i+1} \times {}^{i+1} \rho_{i+1} \right) - m_i \, {}^i \rho_i^* \times {}^i g \qquad \text{(Eq.4.78)} $$



Cette figure schématise la représentation aux niveaux des points O_i, O_{i+1} et G_i de l'ensemble des forces et couples exercés sur un élément du système articulé par les éléments qui lui sont adjacent dans la chaîne cinématique.

Fig. 4.6: Bilan des efforts exercés par les corps adjacents à un élément

4.6.4 Algorithme de "propagation" des forces et couples articulaires

Maintenant, pour alléger les notations et optimiser les calculs du processus itératif, considérons le torseur $\mathcal{F}_i$ tel que:

$$\mathcal{F}_i = \mathcal{F}_i^e - m_i \mathcal{G}_i \qquad \text{(Eq.4.79)}$$

il en résulte que la relation (Eq.4.75) devient:

$${}^i\mathcal{T}_i^{i-1} = {}^i\mathcal{F}_i + \boldsymbol{D}_{i+1*}^i \, {}^{i+1}\mathcal{T}_{i+1}^i \qquad \text{(Eq.4.80)}$$

Posons alors ${}^i\mathcal{F}_i = {}^iF_i + \varepsilon\, {}^iN_i$ relativement à la famille $\mathfrak{f}_i$ et notons ${}^iW_i = {}^i\dot{v}_i - {}^ig$. La séparation des parties réelle et duale de l'équation (Eq.4.80) compte tenu des relations (Eq.4.71), (Eq.4.72), (Eq.4.77) et (Eq.4.78) permet de déduire l'algorithme suivant:

Algorithme 4.6

- **Calcul intermédiaire:**

$$\begin{aligned} {}^iF_i &= m_i \left({}^iW_i + {}^i\dot{\omega}_i \times {}^i\rho_i^\star + {}^i\omega_i \times \left({}^iv_i + {}^i\omega_i \times {}^i\rho_i^\star \right) \right) \\ {}^iN_i &= {}^i\mathbb{I}_i \, {}^i\dot{\omega}_i + {}^i\omega_i \times {}^i\mathbb{I}_i \, {}^i\omega_i + {}^i\rho_i^\star \times {}^iF_i \end{aligned} \qquad \text{(Eq.4.81)}$$

- **Récurrence inverse:**

$$
\begin{aligned}
{}^{i}f_i &= {}^{i}F_i + D^{i}_{i+1}\,{}^{i+1}f_{i+1} \\
{}^{i}n_i &= {}^{i}N_i + D^{i}_{i+1}\left({}^{i+1}n_{i+1} + {}^{i+1}f_{i+1} \times {}^{i+1}\rho_{i+1}\right)
\end{aligned}
$$

4.6.5 Efforts généralisés – Cœfficient caractéristique

- **Cas des articulations cylindriques:**

Dans le cas d'une paire de type C, le mouvement s'effectue avec deux degrés de liberté suivant un même axe, les efforts généralisés correspondant à ces deux degrés de liberté sont donnés par les parties réelle et duale du produit scalaire dual:

$$\{{}^{i}\mathcal{T}_i^{i-1} / {}^{i}\zeta_i\} = \kappa_i + \varepsilon\,\tau_i \qquad \text{(Eq.4.82)}$$

cela s'interprète par l'adjonction d'une force généralisée donnée par:

$$\kappa_i = f_i.z_i \qquad \text{(Eq.4.83)}$$

qui résulte du degré de liberté en translation au couple généralisé au niveau de l'actionneur d'indice i:

$$\tau_i = n_i.z_i \qquad \text{(Eq.4.84)}$$

qui résulte du degré de liberté en rotation.

- **Cas des articulations prismatiques et pivots:**

Dans, le cas où le mécanisme ne comporte que des articulations de type prismatique ou pivot, pour pouvoir déterminer les efforts généralisés au niveau des actionneurs de façon complètement systématique, nous allons introduire un cœfficient dual qui nous permettra de caractériser le type de l'articulation en cours de traitement dans le processus itératif du parcours des actionneurs du système articulé.

Définition 4.5
Nous appelons cœfficient caractéristique du type de l'articulation d'indice i d'un mécanisme ne comportant que des articulations prismatiques ou pivots le nombre dual:

$$\nu_i = \sigma_i + \varepsilon\,\overline{\sigma}_i \qquad \text{(Eq.4.85)}$$

où $(\sigma_i, \overline{\sigma}_i)$ est la partition binaire de l'unité associée à l'articulation d'indice i.

Utilisons là encore une notation unifiée Q_i pour représenter aussi bien les forces généralisées que les couples généralisés qu'on appellera tout simplement effort généralisé.

Par définition, l'effort généralisé appliqué à l'actionneur i est égal à la composante du champ ${}^{i}\mathcal{T}_i^{i-1}$ suivant l'axe (ou suivant le vecteur lorsqu'il s'agit d'une articulation prismatique) défini par le mouvement. C'est à dire:

$$Q_i = [{}^{i}\mathcal{T}_i^{i-1} / \nu_i\,{}^{i}\zeta_i] \qquad \text{(Eq.4.86)}$$

donc, $Q_i = \sigma_i\, n_i.z_i + \overline{\sigma}_i\, f_i.z_i$, telle que:

$$Q_i = \begin{cases} f_i.z_i & \text{si l'articulation } i \text{ est prismatique} \\ n_i.z_i & \text{si l'articulation } i \text{ est de type pivot} \end{cases}$$

De façon générale, on écrira la relation (Eq.4.86) sous la forme:

$$Q_i = (\sigma_i\, n_i + \overline{\sigma}_i\, f_i).z_i \qquad \text{(Eq.4.87)}$$

4.6.6 Algorithme itératif du modèle dynamique

Dans ce paragraphe, nous donnons l'algorithme complet pour la formation du modèle dynamique itératif des systèmes mécaniques articulés à structure de chaîne cinématique ouverte simple. Il inclut les algorithmes 4.5, 4.6 et la relation (Eq.4.87).

Algorithme 4.7

- *Pour i variant de 0 à N − 1 faire:*

$$\begin{aligned}
{}^{i+1}\omega_i &= D_i^{i+1}\,{}^i\omega_i \\
{}^{i+1}\omega_{i+1} &= {}^{i+1}\omega_i + \sigma_{i+1}\,\dot{q}_{i+1}\,{}^{i+1}z_{i+1} \\
{}^{i+1}v_i &= D_i^{i+1}\,{}^i v_i \\
{}^{i+1}v_{i+1} &= {}^{i+1}v_i + {}^{i+1}\omega_i \times {}^{i+1}\rho_{i+1} + \overline{\sigma}_{i+1}\,\dot{q}_{i+1}\,{}^{i+1}z_{i+1} \\
{}^{i+1}\dot{\omega}_i &= D_i^{i+1}\,{}^i\dot{\omega}_i \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}\dot{\omega}_i + \sigma_{i+1}\left({}^{i+1}\omega_{i+1} \times \dot{q}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{q}_{i+1}\,{}^{i+1}z_{i+1}\right) \\
{}^{i+1}W_i &= D_i^{i+1}\,{}^i W_i \\
{}^{i+1}W_{i+1} &= {}^{i+1}W_i + {}^{i+1}\dot{\omega}_i \times {}^{i+1}\rho_{i+1} + \sigma_{i+1}\,{}^{i+1}v_{i+1} \times \dot{q}_{i+1}\,{}^{i+1}z_{i+1} + \\
&\quad \overline{\sigma}_{i+1}\left({}^{i+1}\omega_{i+1} \times \dot{q}_{i+1}\,{}^{i+1}z_{i+1} + \ddot{q}_{i+1}\,{}^{i+1}z_{i+1}\right) \\
{}^{i+1}F_{i+1} &= m_{i+1}\left({}^{i+1}W_{i+1} + {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}\rho^*_{i+1} + {}^{i+1}\omega_{i+1} \times \left({}^{i+1}v_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{i+1}\rho^*_{i+1}\right)\right) \\
{}^{i+1}N_{i+1} &= {}^{i+1}\mathbb{I}_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{i+1}\mathbb{I}_{i+1}\,{}^{i+1}\omega_{i+1} + {}^{i+1}\rho^*_{i+1} \times {}^{i+1}F_{i+1}
\end{aligned}$$

fin faire.

- *Pour i variant de N à 1 faire:*

$$\begin{aligned}
{}^i f_i &= {}^i F_i + D_{i+1}^i\,{}^{i+1}f_{i+1} \\
{}^i n_i &= {}^i N_i + D_{i+1}^i\left({}^{i+1}n_{i+1} + {}^{i+1}f_{i+1} \times {}^{i+1}\rho_{i+1}\right) \\
Q_i &= (\sigma_i\, n_i + \overline{\sigma}_i\, f_i).z_i
\end{aligned}$$

fin faire.

4.6.7 Initialisation des récurrences

Dans le cas où le robot manipulateur est lié à un socle fixe, on peut initialiser la récurrence directe de l'algorithme 4.7 en prenant:

$$^0\omega_0 = \vec{0}, \quad ^0\dot{\omega}_0 = \vec{0}, \quad ^0v_0 = \vec{0}$$

Quant à la quantité W_i, elle peut être initialisée à:

$$^0W_0 = \quad -^0\vec{g}$$

tel que, pour le calcul extrinsèque, $^0\vec{g}$ désigne le vecteur gravité, dans le repère de travail. En général, dans les problèmes de la robotique, le choix du référentiel associé à l'espace de travail (i.e. la famille de référence f_0) est fait de telle sorte[5] que:

$$^0\vec{g} = \quad [0\,0\,\pm g]^t \qquad \text{(Eq.4.88)}$$

où g est la constante locale de gravité.

Dans le cas où on a des robots mobiles, les quantités $^0\omega_0$, $^0\dot{\omega}_0$, $^0v_0$, $^0\dot{v}_0$ seront initialisées aux vitesses et accélérations du support mobile et:

$$^0W_0 = \quad ^0\dot{v}_0 - ^0\vec{g}$$

Par ailleurs, pour prendre en considération dans l'algorithme 4.7 l'effet relatif à l'action de l'organe terminal sur son environnement (i.e. l'opposé des efforts exercés par le milieu extérieur sur l'organe terminal[6]; par exemple: charges, couples, ... etc), il va suffire d'initialiser la récurrence inverse au rang $N+1$ en considérant $^{N+1}f_{N+1}$ et $^{N+1}n_{N+1}$ comme étant respectivement la force et le couple qu'exerce l'organe terminal sur l'environnement extérieur.

4.7 Complexité du calcul algorithmique

La définition de la complexité qu'on propose ici consiste à compter le nombre maximal d'opérations arithmétiques élémentaires d'additions et de multiplications nécessaires au calcul du modèle (ou de l'algorithme) en fonction du nombre de degrés de liberté du système articulé. Ainsi, dans la suite, nous appellerons ce nombre d'opérations: complexité de calcul, coût de calcul ou encore temps de calcul. Dans le but de la mise en œuvre d'une commande dynamique, de nombreux auteurs ont mis au point plusieurs algorithmes permettant d'obtenir de façon numérique le modèle dynamique [Section 0.3, page 4]. En réalité, ce sont les limitations pratiques liées au temps de calcul qui vont permettre de juger des performances d'un modèle (puisque les différents formalismes de la dynamique conduisent à des équations équivalentes). Dans nombre de cas, les limitations du temps de calcul peuvent s'opposer à l'utilisation d'un modèle pour la commande en

[5] D'autres choix peuvent être faits: $^0\vec{g} = [\pm g\,0\,0]^t$ ou encore $^0\vec{g} = [0\ \pm g\,0]^t$.

[6] D'après la loi de l'action et de la réaction de Newton.

temps réel des structures articulées. Un modèle qui nécessite un temps de calcul gigantesque est proscrit malgré les progrès technologiques étendus des nouveaux calculateurs. Ainsi, des efforts considérables ont été déployés dans le but d'optimiser le coût de calcul des différents modèles.

Déterminons maintenant le coût de calcul de l'algorithme 4.7, et comparons son coût à ceux d'un nombre d'algorithmes de la littérature. Soit N -supérieur ou égal à 3- le nombre de degrés de liberté du robot, pour pouvoir comparer tous les algorithmes étudiés dans ce paragraphe dans les mêmes conditions, *on suppose donnés les tenseurs d'inertie des différents éléments du robot en leurs centres de masse*[7], le tableau 4.2 donne une comparaison en complexités du calcul algorithmique des modèles décrits dans Uicker [UICK-65]/Kahn [KAHN-69], Raibert & Horn [RAIB-78], Waters [WATE-79], Hollerbach [HOLL-80], [LUH—80], Dombre & Khalil [DOMB-88] et Hamlili [HAML-92] (algorithme 4.7). Pour établir le calcul des complexités de ces algorithmes, on va donc compter le nombre maximal[8] des opérations élémentaires "effectives" à chaque étape du calcul en ligne qu'ils induisent.

Ainsi, à titre d'exemple et en supposant que le nombre de degrés de liberté $N \geq 3$:

- Le calcul du nombre d'opérations nécessaires au calcul de D_i^{i+1} est donné par la relation (Eq.3.61) [page 94]. Soit 4 multiplications et 0 additions pour le calcul de D_i^{i+1}. D'où $4N$ multiplications et 0 additions pour le calcul des N matrices de passage D_i^{i+1}.

- Le décompte des opérations associées au calcul des quantités définies dans l'algorithme 4.7 s'obtient directement des expressions qui permettent leur calcul. Cependant, dans ce décompte, la multiplication par σ_i et $\overline{\sigma_i}$ ne sont pas réellement des multiplications puisqu'elles vont se traduire au moment de la programmation par des tests conditionnels de la forme:

  si $\cdots$ alors $\cdots$
  sinon $\cdots$

  et donc seules les opérations de la séquence d'instructions sélectionnée pourront être effectuées. De même, toutes les expressions de la forme $a^k z_k$ représentent, en fait, des vecteurs de la forme $[0, 0, a]^T$ et n'induisent donc pas d'opérations effectives.

- Le calcul des quantités Q_i ne nécessite aucune opération, puisqu'il s'agit de poser Q_i comme étant la troisième coordonnée de ${}^i f_i$ ou ${}^i n_i$ selon que la nature de l'articulation d'indice i est de type prismatique ou rotoïde.

[7] Dans plusieurs cas pratiques, certains calculs peuvent être fait hors ligne. Cela permet de diminuer substantiellement la complexité des algorithmes. Ainsi, dans le cas où le mécanisme ne comporte que des articulations rotoïdes, le calcul hors ligne des tenseurs d'inertie aux points O_i induit une amélioration de la complexité des algorithmes basés sur le formalisme de Newton-Euler de $21N$ multiplications et $12N$ additions (ces calcul pouvant être effectués hors ligne dans un module à part). A titre d'exemple: la complexité de notre algorithme n'est plus que $93N - 85$ multiplications et $76N - 72$ additions celle de l'algorithme [DOMB-88] n'est plus que $127N - 47$ multiplications et $89N - 33$ additions [Dombre & Khalil [DOMB-88]].

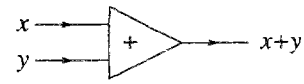
[8] Cas où toutes les articulations du mécanisme sont supposées de type pivot.

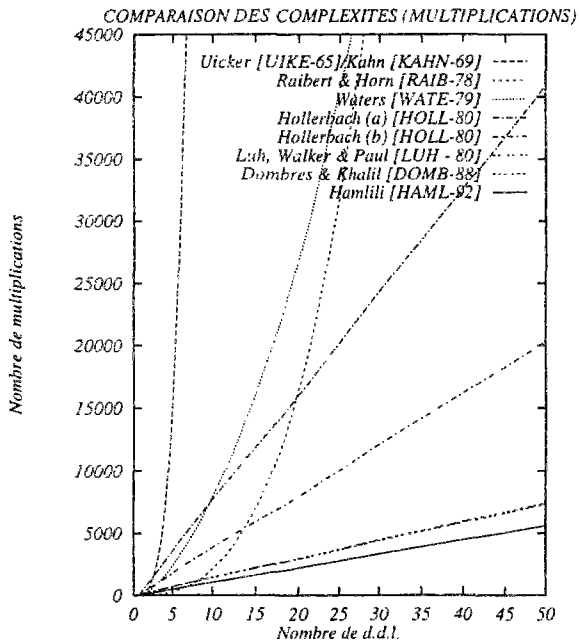
Tableau 4.1 : *Complexité de l'algorithme*		
Quantité	Multiplications	Additions
${}^{i+1}D_i$	$4N$	0
${}^{i+1}\omega_{i+1}$	$8N-6$	$6N-8$
${}^{i+1}v_{i+1}$	$14N-15$	$11N-19$
${}^{i+1}\dot{\omega}_{i+1}$	$10N-15$	$8N-10$
${}^{i+1}W_{i+1}$	$11N-18$	$8N-11$
${}^{i+1}F_{i+1}$	$21N-18$	$18N-15$
${}^{i+1}N_{i+1}$	$24N-13$	$15N-9$
${}^{i}f_i$	$8N$	$8N$
${}^{i}n_i$	$14N$	$14N$
Total	$114N-85$	$88N-72$

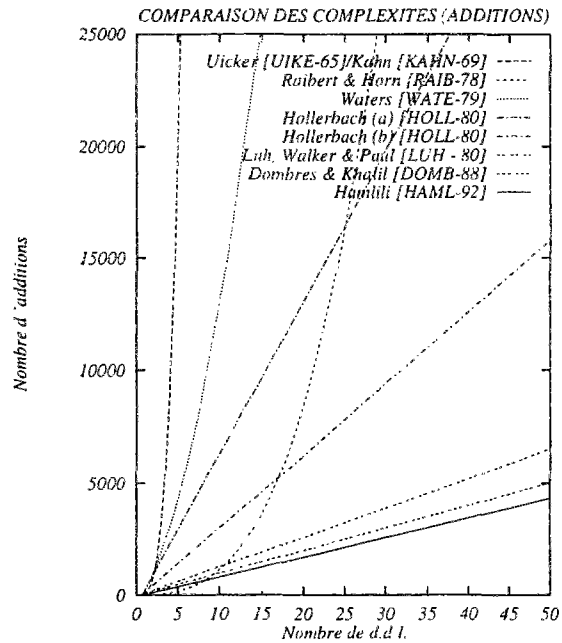
Le tableau 4.1 résume le nombre d'opérations nécessaires aux calcul de ces différentes quantités. Le nombre maximum des opérations élémentaires de l'algorithme 4.7 est donc de: $114N-85$ multiplications et $88N-72$ additions.

Tableau 4.2 : *Comparaison des complexités dans le cas général*			
Méthode	Formulation	Multiplications	Additions
Uicker [UICK-65]/ Kahn [KAHN-69]	dynamique lagrangienne	$16N^4+\frac{215}{6}N^3+\frac{171}{4}N^2+\frac{53}{3}N-128$	$25N^4+22N^3+\frac{129}{2}N^2+\frac{42}{3}N-96$
Raibert & Horn [RAIB-78]	dynamique lagrangienne	$2N^3+N^2$	N^3+N^2+2N
Waters [WATE-79]	dynamique lagrangienne	$53N^2+310N-512$	$82N^2+514N-384$
Hollerbach (a) [HOLL-80]	dynamique lagrangienne, récurrente, (4×4)	$830N-592$	$675N-464$
Hollerbach (b) [HOLL-80]	dynamique lagrangienne, récurrente, (3×3)	$412N-277$	$320N-201$
Luh, Walker & Paul [LUH—80]	dynamique de Newton-Euler, récurrente, (3×3)	$150N-48$	$131N-48$
Dombre & Khalil [DOMB-88]	dynamique de Newton-Euler, récurrente, (3×3)	$148N-47$	$101N-33$
Hamlili [HAML-92]	dynamique de Newton-Euler, récurrente, (3×3)	$114N-85$	$88N-72$

Bien que les deux formalismes de Newton–Euler et de Lagrange donnent lieu à des équations équivalentes qui aboutissent au même résultat à l'erreur de l'arrondi prés du calcul à la machine, on remarque que les méthodes basées sur le formalisme de Newton–Euler sont généralement moins coûteuses. Ainsi, lorsque le nombre N est inférieur ou égal 6, on remarque que notre algorithme est jusqu'à 100 fois plus efficace que certaines formulations basées sur la dynamique lagrangienne [Tableau 4.3]. Toutefois, la méthode de Raibert & Horn [RAIB-78] est moins coûteuse lorsque N demeure inférieur à 6. Cela vient du fait que le cœfficient du monôme de plus haut degré en N des polynômes qui représentent la complexité de calcul de ce modèle, est relativement faible







Cette figure donne les comportements asymptotiques des fonctions de coût des modèles comparés:

- Lorsque le nombre de degrés de liberté croit, on voit bien que le coût en nombre d'opérations élémentaires des méthodes de Uicker/Kahn, Raibert & Horn et celle de Waters grimpe très rapidement pour atteindre des valeurs gigantesques.
- Le nombre de multiplications dans les méthodes de Luh, Walker & Paul et Dombre & Khalil est pratiquement le même.
- Notre algorithme demeure le plus économique de tous.

Fig. 4.7: Comparaison des comportements asymptotiques des modèles

[Tableau 4.2].

Tableau 4.3 : *Le nombre de d.d.l.* $N = 6$		
Méthode	*Multiplications*	*Additions*
Uicker [UICK-65]/ Kahn [KAHN-69]	66 271	51 548
Raibert & Horn [RAIB-78]	468	264
Waters [WATE-79]	7 051	5 652
Hollerbach (a) [HOLL-80]	4 388	3 586
Hollerbach (b) [HOLL-80]	2 195	1 719
Luh, Walker & Paul [LUH—80]	852	738
Dombre & Khalil [DOMB-88]	841	573
Hamlili [HAML-92]	599	456

Ce rapport d'efficacité grimpe jusqu'à 200 fois lorsque le nombre de degré de liberté N est égal à 10 [Fig. 4.7, Tableau 4.4]. A l'exception des algorithmes de Hollerbach (a) et (b) [HOLL-80], les formulations basées sur le formalisme de Lagrange ne peuvent apporter une solution simple aux problèmes de la commande dynamique des robots en temps réel.

Tableau 4.4 : *Le nombre de d.d.l.* $N = 10$		
Méthode	*Multiplications*	*Additions*
Uicker [UICK-65]/ Kahn [KAHN-69]	200 157	278 494
Raibert & Horn [RAIB-78]	2 100	1 120
Waters [WATE-79]	7 888	12 956
Hollerbach (a) [HOLL-80]	7 708	6 286
Hollerbach (b) [HOLL-80]	3 843	2 999
Luh, Walker & Paul [LUH—80]	1 452	1 262
Dombre & Khalil [DOMB-88]	1 433	977
Hamlili [HAML-92]	1 055	808

Tous ces résultats et remarques permettent de conclure que ce sont des méthodes récurrentes telles que [LUH—80], [DOMB-88] ou [HAML-92] qui vont fournir une solution économique au problème de la commande en temps réel. Enfin, ils mettent en valeur les idées sous-jacentes qui nous ont permis de construire notre algorithme, puisqu'il demeure le plus efficace de tous lorsque le nombre de degré de liberté est assez grand.

4.8 Modèle dynamique des mécanismes à structure arborescente

Dans ce paragraphe, on généralise l'algorithme 4.7 à des robots et mécanismes à structure d'arbre topologique. Nous avons vu au chapitre précédent que dans les systèmes articulés à chaîne arborescente, une fois fixé le corps de base, le parcours sans retour en arrière de l'arbre topologique à partir du corps de base dans le sens des organes terminaux révèle l'existence d'un unique antécédent et éventuellement plusieurs successeurs immédiats pour chaque corps du système mécanique -ou sommet du graphe associé- (excepté, bien sûr, le corps de base -la racine de l'arbre graphique- qui ne peut avoir d'antécédent immédiat).

4.8.1 Equations du mouvement d'une structure d'arbre topologique

Dans le cas où le système de solides articulés est à topologie d'arbre, la partie du calcul décrite par la récurrence directe de l'algorithme de "propagation" des grandeurs cinématiques et dynamiques n'est pas plus compliqué que dans le cas des structures à chaîne ouverte simple. Il découle de l'algorithme 4.7 en posant:

Algorithme 4.8

$$
\begin{aligned}
j &= i^-(k) \\
{}^k\omega_j &= D_j^k\, {}^j\omega_j \\
{}^k\omega_k &= {}^k\omega_j + \sigma_k \dot{q}_k\, {}^k z_k \\
{}^k v_j &= D_j^k\, {}^j v_j \\
{}^k v_k &= {}^k v_j + {}^k\omega_j \times {}^k\rho_k + \overline{\sigma}_k \dot{q}_k\, {}^k z_k \\
{}^k\dot{\omega}_j &= D_j^k\, {}^j\dot{\omega}_j \\
{}^k\dot{\omega}_k &= {}^k\dot{\omega}_j + \sigma_k \left({}^k\omega_k \times \dot{q}_k\, {}^k z_k + \ddot{q}_k\, {}^k z_k \right) \\
{}^k W_j &= D_j^k\, {}^j W_j \\
{}^k W_k &= {}^k W_j + {}^k\dot{\omega}_j \times {}^k\rho_k + \sigma_k\, {}^k v_k \times \dot{q}_k\, {}^k z_k + \overline{\sigma}_k \left({}^k\omega_k \times \dot{q}_k\, {}^k z_k + \ddot{q}_k\, {}^k z_k \right) \\
{}^k F_k &= m_k \left({}^k W_k + {}^k\dot{\omega}_k \times {}^k\rho_k^* + {}^k\omega_k \times \left({}^k v_k + {}^k\omega_k \times {}^k\rho_k^* \right) \right) \\
{}^k N_k &= {}^k\mathbb{I}_k\, {}^k\dot{\omega}_k + {}^k\omega_k \times {}^k\mathbb{I}_k\, {}^k\omega_k + {}^k\rho_k^* \times {}^k F_k
\end{aligned}
$$

C'est dans la partie calculée par une récurrence inverse que les expressions du modèle dynamique itératif d'une structure arborescente ouverte diffèrent de celles de l'algorithme 4.7 établi pour les structures en chaîne ouverte simple. En effet, le bilan des efforts exercés sur un élément doit prendre en compte le fait qu'un corps peut admettre plusieurs successeurs immédiats à la fois. La modification de cette partie de l'algorithme est obtenue de la manière suivante:

Algorithme 4.9

$$
\begin{aligned}
{}^j f_j &= {}^j F_j + \sum_{k \in i^+(j)} D_k^j\, {}^k f_k \\
{}^j n_j &= {}^j N_j + \sum_{k \in i^+(j)} D_k^j \left({}^k n_k + {}^k f_k \times {}^k\rho_k \right) \\
Q_j &= (\sigma_j\, n_j + \overline{\sigma}_j\, f_j).z_j
\end{aligned}
$$

Enfin, l'ordre de parcours déterminé par la définition 3.6 (parcours en largeur de l'arbre topologique) permet d'énoncer l'algorithme de calcul du modèle dynamique -des mécanismes à structure d'arbre topologique- sous forme itérative:

Algorithme 4.10

- *pour k variant de 1 à N faire:*

$$
\begin{aligned}
j &= i^-(k) \\
{}^k\omega_j &= D_j^{k\,j}\omega_j \\
{}^k\omega_k &= {}^k\omega_j + \sigma_k\,\dot{q}_k\,{}^k z_k \\
{}^k v_j &= D_j^{k\,j} v_j \\
{}^k v_k &= {}^k v_j + {}^k\omega_j \times {}^k\rho_k + \overline{\sigma}_k\,\dot{q}_k\,{}^k z_k \\
{}^k\dot{\omega}_j &= D_j^{k\,j}\dot{\omega}_j \\
{}^k\dot{\omega}_k &= {}^k\dot{\omega}_j + \sigma_k\,\left({}^k\omega_k \times \dot{q}_k\,{}^k z_k + \ddot{q}_k\,{}^k z_k\right) \\
{}^k W_j &= D_j^{k\,j} W_j \\
{}^k W_k &= {}^k W_j + {}^k\dot{\omega}_j \times {}^k\rho_k + \sigma_k\,{}^k v_k \times \dot{q}_k\,{}^k z_k + \overline{\sigma}_k\,\left({}^k\omega_k \times \dot{q}_k\,{}^k z_k + \ddot{q}_k\,{}^k z_k\right) \\
{}^k F_k &= m_k\,\left({}^k W_k + {}^k\dot{\omega}_k \times {}^k\rho_k^\star + {}^k\omega_k \times \left({}^k v_k + {}^k\omega_k \times {}^k\rho_k^\star\right)\right) \\
{}^k N_k &= {}^k\mathbb{I}_k\,{}^k\dot{\omega}_k + {}^k\omega_k \times {}^k\mathbb{I}_k\,{}^k\omega_k + {}^k\rho_k^\star \times {}^k F_k
\end{aligned}
$$

  fin faire.

- *pour j variant de N à 1 faire:*

$$
\begin{aligned}
{}^j f_j &= {}^j F_j + \sum_{k\in i^+(j)} D_k^{j\,k} f_k \\
{}^j n_j &= {}^j N_j + \sum_{k\in i^+(j)} D_k^j\,\left({}^k n_k + {}^k f_k \times {}^k\rho_k\right) \\
Q_j &= (\sigma_j\, n_j + \overline{\sigma}_j\, f_j).z_j
\end{aligned}
$$

  fin faire.

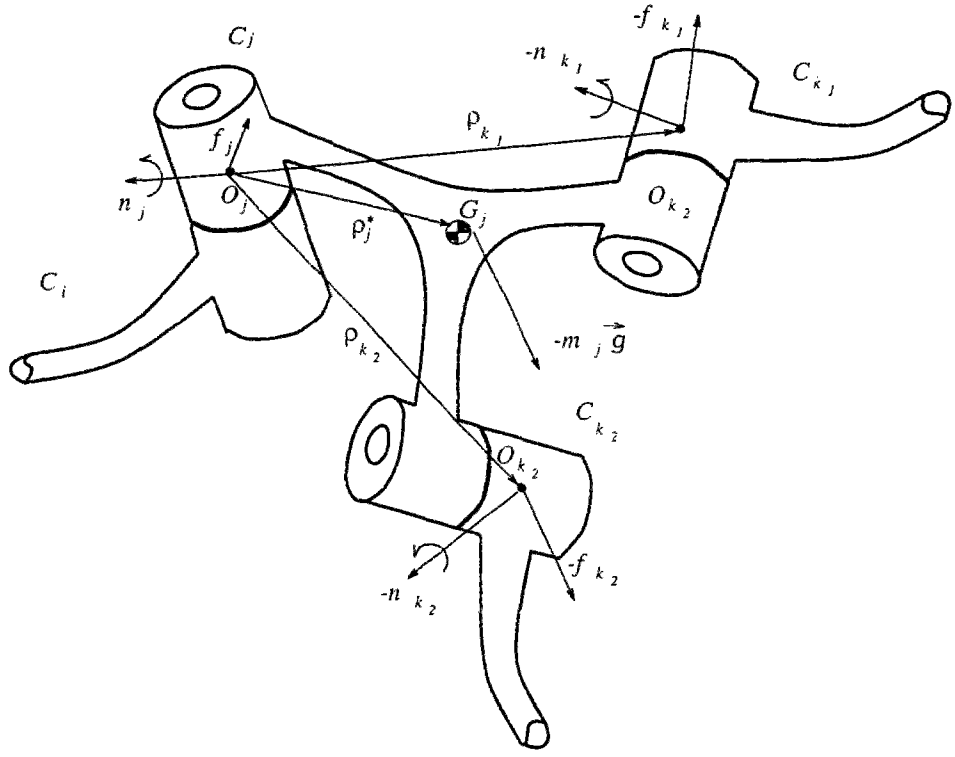
Cet algorithme résume les deux algorithmes 4.8 et 4.9.

4.8.2 Initialisation de l'algorithme itératif

Comme dans le paragraphe 4.6.7, pour initialiser l'algorithme 4.10, il faut:

- D'une part, initialiser la récurrence directe en fixant les vecteurs ${}^0\omega_0$, ${}^0\dot{\omega}_0$, ${}^0v_0$ et ${}^0\dot{v}_0$ selon les données du problème à traiter (Exemples: repère terrestre ou non, corps de référence en mouvement ou fixe, ...etc.) et poser le cas échéant: ${}^0W_0 = {}^0\dot{v}_0 - {}^0\vec{g}$ (${}^0\vec{g} = 0$ dans le cas d'absence de champ de gravitation).

- D'autre part, initialiser la récurrence inverse en fixant les les valeurs des efforts exercés par les organes terminaux sur leur(s) environnement(s).



Cette figure schématise les efforts exercés sur l'élément C_j par les éléments adjacents dans l'arborescence de la structure topologique du mécanisme. Ce bilan des efforts doit prendre en considération le fait qu'un corps peut avoir plusieurs successeurs immédiats.

Fig. 4.8: Bilan des efforts entre corps adjacents dans une structure arborescente

4.9 Conclusion

Dans ce chapitre, nous avons tout d'abord défini la notion de dérivée de champ [Paragraphe 4.2] et celle de la représentation adjointe de mouvements [Paragraphe 4.3]. Ces définitions nous ont permis de définir la notion de champ de pseudo-accélération [Définition 4.3] et la loi de composition de tels champs. L'expression explicite de l'opérateur généralisé d'inertie que nous avons établi, au paragraphe 4.4, nous a permis de regrouper le principe fondamentale de la dynamique et le théorème du moment cinétique dans une même formule synthétique [Paragraphe 4.5]. Nous avons ensuite donné une structuration itérative complète du modèle dynamique des mécanismes à

chaînes ouvertes simples, au moyen de la théorie des groupes et algèbres de Lie. Cette formulation est intrinsèque et elle est applicable aussi dans le cas où les mécanismes présentent des articulations de type quelconque et particulièrement dans le cas des articulations de type C. Les mécanismes dont les articulations sont de type prismatique ou pivot sont de loin les plus nombreux[9]. Aussi, la formulation des modèles en prenant compte de cette hypothèse revêt-elle une importance particulière. En introduisant les notions de partitions binaires de l'unité et du cœfficient caractéristique dual d'une articulation nous avons établi un algorithme efficace (de faible coût) pour le calcul du modèle dynamique des systèmes articulés à chaînes ouvertes simples dont les articulations sont de types rotoïdes ou prismatiques [Paragraphe 4.6.6, Algorithme 4.7]. La comparaison de la complexité de notre algorithme à sept autres algorithmes de référence a révélé que notre algorithme demeure celui qui nécessite le plus faible coût de calcul. Ce qui implique qu'il est très adapté à être intégré dans une commande en temps réel.

Enfin, grâce à la théorie des graphes et à l'introduction d'un bon ordre de parcours de l'arbre topologique des systèmes mécaniques arborescents, nous avons généralisé notre modèle dynamique itératif pour recouvrir une large catégorie de mécanismes [Paragraphe 4.8.1, Algorithme 4.10]. L'ordre de parcours des éléments d'un robot ou un système articulé à chaîne arborescente que nous avons introduit dans le chapitre précédent permet, alors, l'automatisation du processus itératif pour le calcul du modèle dynamique de telles structures.

[9]Même dans le cas contraire, on peut toujours se ramener à des modèles de ce type.

Partie 3:

CONCEPTION ET REALISATIONS INFORMATIQUES

– *Résumé de la partie 3* –

LES SYSTEMES *SURVEYOR* ET *MEDUSA MF 77*

Dans cette troisième partie, nous présentons le système SURVEYOR, notre prototype de système de calcul formel basé sur la réécriture typée et les techniques de l'intelligences artificielle. Il s'avérera un puissant outil de transformation syntaxique des expressions mathématiques dans des spécifications équationnelles et le logiciel MEDUSA MF77 développé dans le but de générer automatiquement des schémas quasi-optimaux pour le calcul des équations de la dynamique des systèmes articulés. Ainsi, nous considérons les problèmes relatifs à la génération d'un code optimisé du modèle dynamique itératif, nous allons montrer sur un exemple simple que les fonctionnalités actuelles des systèmes de calcul formel classiques ne permettent pas la production de code qui requièrent les facteurs souhaités de qualité.

Nous exposerons ensuite les solutions que nous apportons à ces problèmes par la réécriture et les techniques de l'intelligence artificielle. Ainsi, nous allons introduire la faculté de suspendre momentanément des informations au toplevel du logiciel de calcul formel pour laisser aller à son terme le processus de génération du code source quasi-optimal décrivant ainsi un schéma de calcul itératif.

Chapitre 5

LE PROTOTYPE: *SURVEYOR*

5.1 Introduction

L'informatique se définit comme l'ensemble des disciplines scientifiques et des techniques spécifiquement applicables au traitement de l'information, notamment par des moyens automatiques. L'intelligence artificielle est une discipline très avancée de l'informatique dont elle utilise les techniques pour essayer de réaliser des automates adoptant une démarche proche de la pensée humaine. Ainsi, elle concerne l'ensemble des modélisations de la connaissance et des techniques permettant d'atteindre ce but. En informatique traditionnelle, les programmes traitent essentiellement de l'information numérique; donc, une information quantitative. Par contre, les programmes de l'intelligence artificielle traitent essentiellement les informations de nature symbolique et qualitative. Une des raisons qui ont poussées au développement des programmes de l'intelligence artificielle est celle de concevoir des machines qui permettent à l'utilisateur de déclarer des faits plutôt que des ordres. L'outil essentiel dans de tels développements est donc la notion ensembliste de relations [J.P. Sansonnet [SANS-85]].

Citons ici J. Pitrat "*Presque tout reste encore à faire en intelligence artificielle*" [J. Pitrat [PITR-90]]. En effet, malgré les efforts considérables qui ont été déployés dans la recherche dans ce domaine et l'immense succès rapporté dans la conception et le développement de logiciels généraux et donc "réutilisables", il reste encore beaucoup à faire pour améliorer les performances de ces systèmes et bien que l'on soit parvenu dans les tarvaux actuels à décrire des procédés pour améliorer les performances d'un certain nombre d'applications, on est encore incapable de pouvoir les généraliser par des méthodes universelles "réutilisables" dans toute autre application. Ces difficultés sont particulièrement ressenties par des chercheurs qui ont réalisé des logiciels capables de résoudre une grande classe de problèmes [J.L. Laurière [LAUR-76], D. Fieschi [FIES–84]].

Le système *SURVEYOR* est un prototype de système algébrique basé sur la réécriture typée[1] des expressions formelles. Il permet de traiter des expressions mathématiques par réécriture à partir d'une description déclarative (et/ou semi-déclarative) exprimée par une ou plusieurs spécifications équationnelles (i.e. réécriture au moyen d'un ou plusieurs systèmes d'équations). Ainsi, l'objectif général du projet *SURVEYOR* est d'étudier, concevoir et tester des mécanismes et des outils permettant la réduction systématique d'expressions mathématiques à partir de spécifications axiomatiques équationnelles. Son implémentation était faite dans la perspective de répondre à des questions dont la difficulté se révèle avec l'étude de points délicats concernant les problèmes d'évaluation, d'unification, de transformation symbolique, de suspension et de restauration d'informations au toplevel d'un interpréteur symbolique. Ainsi, ce chapitre est dédié à décrire les idées qui nous ont permis de réaliser *SURVEYOR* et les solutions proposées pour la résolution d'une large catégorie de problèmes où le contrôle de la connaissance peut se faire par des spécifications équationnelles.

5.2 Conception et prototypage

Il existe différentes méthodes pour concevoir un logiciel. Ces méthodes se basent en général sur un ensemble de recherches théoriques et pratiques. Cependant, l'ensemble des recherches menées dans ce domaine mettent en relief trois étapes essentielles:

- Etape de réflexion et d'analyse du problème: elle consiste à définir les fonctions, les tâches et l'ensemble des services du système en fonction des objectifs initiaux d'une part et des objectifs et besoins découverts au cours du développement d'autre part.
- Etape de réalisation: pour réaliser un système informatique, il faut choisir un langage de programmation en fonction des besoins de conception.
- Etape d'évaluation et de critique des solutions: une critique défavorable peut amener le concepteur à une nouvelle analyse et un nouveau développement pour améliorer la solution ou pour combler ses défaillances du système.

On imagine bien que ces trois étapes sont plus ou moins parallèles et vont coexister jusqu'à l'élaboration complète ou partielle du produit informatique.

Pour concevoir un logiciel qui répond à un nombre de critères de qualité, on passe souvent par un prototypage rapide. En général, l'ensemble des besoins et objectifs pour construire un système -qui n'existe pas encore- sont exprimés dans le langage naturel. Par ailleurs, il est souvent difficile et voir même impossible, d'avoir à l'avance une idée complète de ce que sera le système après sa réalisation. On ne pourra le constater qu'une fois le produit fini et testé au bout d'une certaine période. Ainsi, le développement doit être guidé non seulement par des aspects théoriques mais

[1] La réécriture typée telle que nous allons la définir consiste en une généralisation de la réécriture (au sens classique en informatique théorique).

aussi par les problèmes que l'on rencontre au cours de la réalisation. Les avantages que présente cette méthode de conception peuvent être énumérés en six points essentiels:

- on peut donner forme aux objectifs initiaux exigés dans le produit final;
- on peut détecter des objectifs manquants;
- on peut détecter des fonctions manquantes;
- on peut simuler les fonctions difficilement atteignables;
- on peut découvrir des incohérences au cours du développement du prototype;
- le prototype peut servir de spécification pour le développement du produit final.

5.3 Choix du langage de programmation

Nous pensons que le choix d'un système quelconque de calcul formel comme langage de programmation pour réaliser un logiciel permettant la représentation des connaissances algébriques et équationnelles revient à privilégier un mode de communication Homme-Machine. Le développement informatique d'une méthode de représentation des connaissances équationnelles sous une forme déclarative -ou algorithmique- à travers le langage d'un système, aussi puissant soit il, présente toujours l'inconvénient de conditionner la façon de penser du programmeur -dans la résolution d'un problème mathématique donné- aux impératifs de représentation des connaissances dans le système en question. Ainsi, ce choix n'est pas neutre puisqu'il préjuge de l'efficacité de représentation de ces connaissances (définition des types, algorithmique de résolution d'un problème ... etc). En réalité, un prototype peut être écrit dans n'importe quel langage de programmation et, de ce fait, dans les problèmes qui se rattachent à la connaissance, on préfère généralement l'exprimer dans un langage proche du langage naturel et, surtout, un langage qui permettra de modifier facilement les premières ébauches des travaux.

Dans le cas qui nous intéresse, nous avons choisi de prototyper notre système dans le langage Lisp[2]. Ce choix se justifiait, non seulement parce que la solution que nous préconisions concerne la manipulation symbolique de l'information, mais encore, parce que la structure interne de Lisp (basée sur des listes) est très proche de la structure d'arbre (représentation abstraite des termes).

5.4 Modèles de représentation des connaissances

Le cerveau humain "machine à penser" de notre corps est tellement complexe qu'il est impossible de dire avec certitude comment la connaissance y est représentée et stockée ou de décrire les fonctions par lesquelles nous arrivons à exploiter cette connaissance. Ici, nous entendons par connaissance toute forme du savoir humain. Par ailleurs, il est certain et évident que même dans

[2] *SURVEYOR* est écrit dans le dialecte le_lisp (version 15.24).

les tâches intellectuelles les plus simples, l'homme fait appel à des connaissances tellement éparses et composées pour prendre part à une idée, trouver une ressemblance entre deux choses, prendre une décision, démontrer un résultat ...etc. Ce "mouvement de recherche" continu de l'esprit est trop complexe pour être décrit par un modèle universel ou d'une manière fidèle. Cependant, on peut donner des modèles partiels qui imitent ces fonctions dans des cas particuliers. En I.A., il existe différents modèles pour la représentation des connaissances:

- les programmes traditionnels (la représentation procédurale),
- les automates finis,
- les bases de données,
- les réseaux sémantiques et Frames,
- les représentations logiques,
- les systèmes de réécritures,
- les spécifications formelles,
- les règles de production.

Les tendances actuelles en I.A. s'orientent vers la conception de systèmes qui autorisent simultanément différents types de représentions des connaissances et de raisonnements dans le but de mieux recueillir, traduire et traiter la connaissance.

5.5 Programmation traditionnelle

Nous entendons par programmation traditionnelle (anglais: traditional programing): l'ensemble des activités orientées vers la conception, la réalisation, le test et la maintenance des algorithmes écrits dans un langage de programmation donné. La programmation traditionnelle n'est souvent utilisée en I.A. que pour l'écriture du système de contrôle. Mais, elle est rarement utilisée pour représenter des connaissances -souvent déclaratives- vu sa faiblesse à structurer les données et sa lourdeur de mise en œuvre. Cette difficulté vient du fait que ces connaissances ne sont généralement pas suffisamment claires pour être décrites à l'avance.

5.6 Réécriture

La réécriture concerne tous les aspects de la manipulation symbolique de l'information. Elle consiste en la transformation syntaxique des expressions exprimées dans des spécifications équationnelles. Dans la suite nous entendons par spécification équationnelle un ensemble d'axiomes équationnels.

Exemple

La spécification équationnelle décrivant les calculs dans un groupe G, dont la loi de composition interne "$*$" est notée multiplicativement et l'élément neutre est noté e, peut être formulée sous la forme de six équations:

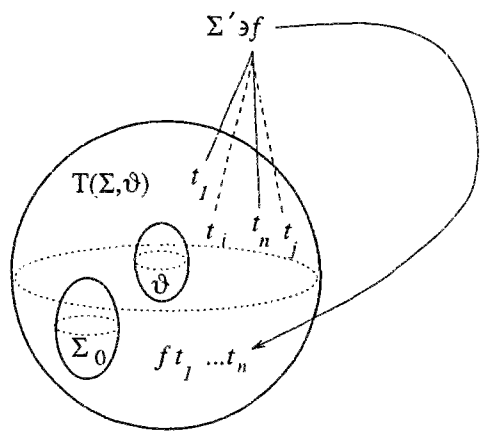
$x * e = x$	;	$e * x = x$
$x * x^{-1} = e$	;	$x^{-1} * x = e$
$(x * y)^{-1} = y^{-1} * x^{-1}$	;	$(x * y) * z = x * (y * z)$

Si la loi "$*$" est commutative, la spécification équationnelle du groupe G devient alors:

$x * e = x$	;	$x * x^{-1} = e$
$(x * y)^{-1} = y^{-1} * x^{-1}$	;	$(x * y) * z = x * (y * z)$
	$x * y = y * x$	

5.6.1 Termes

Pour introduire le cadre formel de la notion des termes du premier ordre nous allons considérer



Cette figure schématise les axiomes i) et ii) qui définissent la construction des termes de $T(\Sigma, \vartheta)$ sur l'alphabet Σ et l'ensemble des variables ϑ. Dans cette figure Σ_0 désigne l'ensemble des constantes (symboles de fonctions d'arité égale à 0) et Σ' désigne l'ensemble des symboles de fonction d'arité non nulle (i.e. $\Sigma = \Sigma' \cup \Sigma_0$ et $\Sigma' \cap \Sigma_0 = \emptyset$).

Fig. 5.1: Construction des termes

deux ensembles:

- Un ensemble Σ de symboles de fonctions.

- Un ensemble ϑ de symboles appelés variables.

et une application $ar : \Sigma \rightarrow \mathbb{N}$. L'image $ar(f)$ d'un symbole de fonction est dit *arité de* f.

Définition 5.1
Tout terme est formé par l'application des deux axiomes suivant:

i) Les constantes et les variables sont des termes.

ii) Si $f \in \Sigma$ est un symbole de fonction d'arité n et si $t_1, t_2, \cdots, t_n$ sont des termes alors $f(t_1, t_2, \cdots, t_n)$ est un terme (on considère ici la notation pré-fixée).

On notera, dans toute la suite de cet exposé, $T(\Sigma, \vartheta)$ l'ensemble des termes construits sur l'alphabet $\Sigma \cup \vartheta$.

5.6.2 Arbre étiqueté – Sous-terme

Soient une application t de $\mathbb{N}^*$ (ici $\mathbb{N}^*$ est considéré comme étant l'ensemble des mots construits sur l'alphabet $\mathbb{N} \setminus \{0\}$ par l'opération de concaténation) dans $\Sigma \cup \vartheta$ et $O(t)$ son domaine de définition. Alors t est appelé arbre étiqueté sur $\Sigma \cup \vartheta$, si et seulement si:

- Le mot vide[3] ϵ appartient à $O(t)$.
- Si p appartient à $O(t)$, alors $p.i$ (produit de concaténation) appartient à $O(t)$, si et seulement si, $1 \leq i \leq ar(t(p))$.

Définition 5.2
Soient un terme t et p une occurrence de t. Le terme issu de t à l'occurrence p (noté $t_{/p}$) est défini par:

- $t_{/\epsilon} = t$,
- $f(t_1, t_2, \cdots, t_n)_{/i.p} = t_{i/p}$.

5.6.3 Substitution – Filtrage – Unification

On peut définir le remplacement du sous-terme à l'occurrence p d'un terme t par un terme t'. De point de vue mécanique (au sens machinal du terme), ce remplacement peut être regardé comme une succession de deux opérations subordonnées:

- effacement du sous-terme $t_{/p}$ de t à l'occurrence p,
- greffage du terme t' à l'occurrence p (occurrence libérée à l'étape précédente).

Cette opération peut être formalisée comme il suit:

Définition 5.3
Considérons deux termes t et t' de $T(\Sigma, \vartheta)$ et p une occurrence de t. par remplacement de son sous-terme à l'occurrence p par t' et que l'on notera dans la suite $t[p \longleftarrow t']$ est défini par:

- $t[\epsilon \longleftarrow t'] = t'$,

[3]Attention, il ne faut pas confondre la calligraphie du mot vide ϵ et celle de l'unité duale ε.

- $f(t_1, t_2, \cdots, t_n)[i.p \longleftarrow t'] = f(t_1, t_2, \cdots, t_i[p \longleftarrow t'], \cdots, t_n)$.

Ainsi, dans la suite, l'ensemble des variables d'un terme $t \in T(\Sigma, \vartheta)$ peut être défini implicitement par:

$$V_t = \{x \in \vartheta \,/\, \exists p \in O(t) \text{ tel que } t_{/p} = x\} \qquad \text{(Eq.5.1)}$$

ou inductivement par:

$$V_t = \begin{cases} \emptyset & \text{si } t \text{ est un terme clos} \\ \{x\} & \text{si } t = x \text{ est une variable de } \vartheta \\ V_{t_1} \cup V_{t_2} \cup \cdots \cup V_{t_n} & \text{si } t = f(t_1, t_2, \cdots, t_n) \end{cases} \qquad \text{(Eq.5.2)}$$

Définition 5.4

On appelle substitution toute application σ de ϑ dans $T(\Sigma, \vartheta)$ telle que son ensemble de définition est défini par: $dom(\sigma) = \{x \in \vartheta / \sigma(x) \neq x\}$ soit fini.

Proposition 5.1

Considérons une substitution σ. Il existe un unique prolongement $\hat{\sigma}$ à $T(\Sigma, \vartheta)$ qui vérifie pour tout symbole de fonction f (avec $n = ar(f)$ et pour toute suite $t_1, t_2, \cdots, t_n$ de termes de $T(\Sigma, \vartheta)$:

$$\hat{\sigma}(f(t_1, t_2, \cdots, t_n)) = f(\hat{\sigma}(t_1), \hat{\sigma}(t_2), \cdots, \hat{\sigma}(t_n))$$

i.e. tel que $\hat{\sigma}$ soit un endomorphisme de $T(\Sigma, \vartheta)$.

▮ Preuve :

Supposons qu'il existe deux prolongements de σ notés $\hat{\sigma}_1$ et $\hat{\sigma}_2$ tels que:

$$\begin{cases} \hat{\sigma}_1(f(t_1, t_2, \cdots, t_n)) = f(\hat{\sigma}_1(t_1), \hat{\sigma}_1(t_2), \cdots, \hat{\sigma}_1(t_n)) \\ \hat{\sigma}_2(f(t_1, t_2, \cdots, t_n)) = f(\hat{\sigma}_2(t_1), \hat{\sigma}_2(t_2), \cdots, \hat{\sigma}_2(t_n)) \end{cases}$$

Il faut montrer que pour tout terme $t \in T(\Sigma, \vartheta)$:

$$\hat{\sigma}_2(t) = \hat{\sigma}_1(t) \qquad \text{(Eq.5.3)}$$

Nous allons montrer ce résultat par récurrence sur la longueur $|t|$ du terme t:

- si t est un terme de longueur $|t| = 1$, alors deux cas se présentent:
  - $t = c$, est une constante de Σ, alors: $\hat{\sigma}_2(c) = c = \hat{\sigma}_1(c)$,
  - $t = x$, est une variable de ϑ, alors: $\hat{\sigma}_2(x) = \sigma(x) = \hat{\sigma}_1(x)$,
- soit $m \geq 1$, supposons que la relation (Eq.5.3) soit vérifiée pour tout terme de longueur $|t| \leq m$ et montrons que cela reste vrai si $|t| = m + 1$. Considérons alors un terme t tel que $|t| = m + 1$, alors il existe un symbole de fonction f et n termes $t_1, t_2, \cdots, t_n$ (où $n = ar(f)$) de $T(\Sigma, \vartheta)$ tels que: $t = f(t_1, t_2, \cdots, t_n)$. Alors: $\hat{\sigma}_2(t) = f(\hat{\sigma}_2(t_1), \hat{\sigma}_2(t_2), \cdots, \hat{\sigma}_2(t_n))$
  or pour tout $i \in [1..n]$: $|t_i| \leq |t| - 1 = m$, donc par hypothèse de récurrence: $\hat{\sigma}_2(t_i) = \hat{\sigma}_1(t_i)$.
  Ce qui montre la relation (Eq.5.3) pour tout terme de longueur $m + 1$.

C.Q.F.D. ∎

Dans la suite aussi bien une substitution σ que son prolongement à $T(\Sigma, \vartheta)$ seront représentés par le même symbole σ. Par ailleurs, l'ensemble V^σ des variables introduites par la substitution σ peut être défini comme la réunion des ensembles de variables des termes obtenus par instanciation de son domaine de définition:

$$V^\sigma = \bigcup_{x \in dom(\sigma)} V_{\sigma(x)} \qquad \text{(Eq.5.4)}$$

Définition 5.5

Soient t et t' deux termes de $T(\Sigma, \vartheta)$. On dira que t filtre t' (i.e. t' est une instance de t), s'il existe une substitution σ telle que: $t' = \sigma(t)$.

La substitution σ ainsi définie s'appelle un filtre de t vers t'.

Proposition 5.2

Soient t et t' deux termes de $T(\Sigma, \vartheta)$ et σ un filtre de t vers t'. Alors:

$$V_{t'} = (V_t \setminus dom(\sigma)) \cup V^\sigma \qquad \text{(Eq.5.5)}$$

Preuve :

Pour montrer l'égalité (Eq.5.5), il suffit de montrer la double-inclusion équivalente:

- D'une part, il est clair que d'après la définition de t': $V_t \setminus dom(\sigma) \subset V_{t'}$ et $V^\sigma \subset V_{t'}$.
- D'autre part, soit $x \in V_{t'}$, alors deux cas:
  - $x \in V_t$, or comme $t' = \sigma(t)$, $x \in V_t \setminus dom(\sigma)$.
  - $x \notin V_t$, alors x provient de la substitution des variables $V_t \cap dom(\sigma)$, i.e. $x \in V^\sigma$.

C.Q.F.D. ∎

Définition 5.6

Soient t et t' deux termes de $T(\Sigma, \vartheta)$. On dira qu'ils sont unifiables, s'il existe une substitution σ telle que: $\sigma(t) = \sigma(t')$.

La substitution σ ainsi définie s'appelle un unificateur des termes t et t'.

5.6.4 Systèmes de réécriture

Lorsque la connaissance relative à une théorie est exprimée par un ensemble d'équations $E = \{g_i = d_i / g_i, d_i \in T(\Sigma, \vartheta),\ i = 1..N\}$, la théorie de réécriture considère ces équations comme des règles de réduction orientées de la forme $g_i \longrightarrow d_i$ ou $d_i \longrightarrow g_i$. L'ensemble R de ces règles de réécriture est alors appelé système de réécriture induit par la spécification équationnelle E.

Définition 5.7

Soient R un système de réécriture et deux termes $t, t' \in T(\Sigma, \vartheta)$. Nous dirons que t se réécrit en t' et on note $t \longrightarrow_R t'$, si il existe une règle $g \longrightarrow d$ de R, une occurrence p de t et une substitution σ telles que:

- *σ filtre g (terme membre-gauche de la règle $g \longrightarrow d$) vers le sous-terme $t_{/p}$ à l'occurrence p de t.*
- *t' soit le résultat du remplacement de son sous-terme à l'occurrence p par l'instance de d (terme membre-droit de la règle de réécriture $g \longrightarrow d$) par σ.*

Lorsqu'un terme t se réécrit en un terme t', on dit aussi que t se réduit en t'. Cela peut s'écrire, sous forme relationnelle, dans le formalisme que nous avons défini jusqu'ici:

$$t_{/p} = \sigma(g) \text{ et } t' = t[p \longleftarrow \sigma(d)]$$

Soient la relation de réécriture $\longrightarrow_R$ et $\longrightarrow_R^*$ sa clôture réflexive-transitive sur $T(\Sigma, \vartheta)$. En général, on dit qu'un terme t se réécrit en t', si: $t \longrightarrow_R^* t'$.

5.6.5 Terminaison et confluence d'un système de réécriture

Pour que la réécriture d'un terme au moyen d'un système de réécriture soit correcte, il faut qu'elle aboutisse à une forme irréductible au bout d'un nombre fini de transformations (propriété de terminaison) et que deux termes égaux dans la spécification équationnelle E se réécrivent en un même troisième terme (propriété de confluence de Church-Rosser).

Définition 5.8

- *Un terme t de $T(\Sigma, \vartheta)$ est dit en forme normal ou irréductible, s'il n'existe pas de terme $t' \in T(\Sigma, \vartheta)$ tel que: $t \longrightarrow_R t'$.*
- *Un terme t' est dit forme normale d'un terme t, si t se réécrit en t' et t' est en forme normale.*

Il est clair que si un terme possède une forme normale, celle-ci n'est pas nécessairement unique.

Définition 5.9

Considérons un système de réécriture R. On dit que R est nœtherien ou que $\longrightarrow_R$ possède la propriété de terminaison fini, si pour tout terme t de $T(\Sigma, \vartheta)$ il n'existe pas de dérivation infinie:

$$t = t_1 \longrightarrow_R t_2 \longrightarrow_R \cdots \longrightarrow_R t_n \longrightarrow_R \cdots$$

Cela signifie que lorsque la relation de réécriture $\longrightarrow_R$ possède la propriété de terminaison fini, tout terme possède au moins une forme normale.

Définition 5.10

Etant donné un système de réécriture, la relation $\longrightarrow_R$ est dite:

- *confluente en t, si:*

$$[\forall t_1, t_2 \in T(\Sigma, \vartheta) : t \longrightarrow_R^* t_1 \text{ et } t \longrightarrow_R^* t_2] \Longrightarrow [\exists t' \in T(\Sigma, \vartheta) : t_1 \longrightarrow_R^* t' \text{ et } t_2 \longrightarrow_R^* t']$$

- *localement confluente en t, si:*

$$[\forall t_1, t_2 \in T(\Sigma, \vartheta) : t \longrightarrow_R t_1 \text{ et } t \longrightarrow_R t_2] \Longrightarrow [\exists t' \in T(\Sigma, \vartheta) : t_1 \longrightarrow_R^* t' \text{ et } t_2 \longrightarrow_R^* t']$$

- *fortement confluente en t, si:*

$$[\forall t_1, t_2 \in T(\Sigma, \vartheta) : t \longrightarrow_R t_1 \text{ et } t \longrightarrow_R t_2] \Longrightarrow [\exists t' \in T(\Sigma, \vartheta) : t_1 \longrightarrow_R t' \text{ et } t_2 \longrightarrow_R t']$$

Un système de réécriture R est dit confluent, localement confluent ou fortement confluent, si la relation $\longrightarrow_R$ associée est respectivement confluente, localement confluente ou fortement confluente pour tout terme de $T(\Sigma, \vartheta)$.

En général, la propriété de confluence n'est pas vérifiée par une représentation équationnelle. En ajoutant de nouvelles équations déduites des règles de réécriture dont les termes gauches se superposent (i.e ils "chevauchent") et en orientant ces équations, la procédure de Knuth-Bendix [D.E. Knuth & P.B. Bendix [KNUT-70]] permet alors de construire dans certains cas un système confluent [J.P. Jouannaud & E. Kounalis [JOUA-86], N. Dershowitz & J.P. Jouannaud [DERS-89]].

5.6.6 Ordre de réduction

Définition 5.11

Soit $\succ$ un ordre sur l'algèbre $T(\Sigma, \vartheta)$. On dit que $\succ$ est un ordre de réduction, s'il possède les propriétés suivantes:

- *$\succ$ est bien fondé. C'est à dire, qu'il n'existe pas une suite infinie de termes telle que:*

$$t_1 \succ t_2 \succ \cdots \succ t_n \succ \cdots$$

- *$\succ$ possède la propriété de compatibilité. i.e.:*

$$\forall f \in \Sigma', \forall t_1, t_2 \in T(\Sigma, \vartheta) : \quad t_1 \succ t_2 \Longrightarrow f(\cdots, t_1, \cdots) \succ f(\cdots, t_2, \cdots)$$

- *$\succ$ est stable par instanciation. i.e. pour toute substitution σ:*

$$\forall t_1, t_2 \in T(\Sigma, \vartheta) : \quad t_1 \succ t_2 \Longrightarrow \sigma(t_1) \succ \sigma(t_2)$$

5.7 Réécriture typée

Dans ce paragraphe, nous allons définir le cadre axiomatique et formel de l'idée de base qui va nous servir pour représenter la connaissance dans le système *SURVEYOR* et le modèle de son calcul symbolique. Ainsi, dans désormais, on ne considère plus une réécriture d'un terme à partir d'un seul système de réécriture, mais à partir d'une succession de systèmes de réécriture.

Définition 5.12

Nous appelons réécriture typée, sur l'algèbre $T(\Sigma, \vartheta)$, la donnée:

- *d'un ensemble fini $\mathfrak{R}$ de N systèmes de réécriture notés $(R_i)_{i=1..N}$,*
- *d'un ensemble $\mathfrak{A} = \{A_1, A_2, \cdots, A_N\}$ de N éléments, dits attributs de réécriture, muni d'une relation d'ordre partiel stricte $<$ avec premier élément; c'est à dire, telle que:*

$$\forall i \in [2..N] : \quad A_1 < A_i$$

- *d'une application d'attribution bijective **att** définie de $\mathfrak{R}$ dans $\mathfrak{A}$.*

D'un point de vue formel, du fait que nous avons supposé l'application ***att*** bijective, nous aurions pu définir la réécriture typé seulement à partir d'un ensemble partiellement ordonné avec premier élément (pour cette relation d'ordre partiel) de systèmes de réécriture. Mais pour des raisons de clarté de l'exposé nous préférons la définir à partir du triplet $\mathfrak{T} = (\mathfrak{R}, \mathfrak{A}, att)$.

Définition 5.13

Nous dirons que le terme t se réécrit en t' pour la réécriture typée $\mathfrak{T} = (\mathfrak{R}, \mathfrak{A}, att)$, et on écrira $t \xrightarrow{\mathfrak{T}} t'$, s'il existe un sous-ensemble $F = \{A_{i_1}, A_{i_2}, \cdots, A_{i_n}\}$ totalement ordonné de $\mathfrak{A}$ (les A_{i_j} sont supposés être donnés dans l'ordre croissant) et n termes $t_1, t_2, \cdots, t_{n+1}$ de $T(\Sigma, \vartheta)$ tels que:

- $A_{i_1} = A_1 \in F$,
- $\forall j \in [1..n] : \quad R_{i_j} = att^{-1}(A_{i_j})$,
- $t_1 = t$ *et* $t_{n+1} = t'$,
- $\forall j \in [1..n] : \quad t_j \longrightarrow^*_{R_{i_j}} t_{j+1}$.

Lemme 5.3

Dans le cas ou $\mathfrak{R}$ est un singleton, on retrouve la réécriture au sens classique.

Preuve :

En effet, dans ce cas précis F est également un singleton et $F = \mathfrak{R}$. ∎

La réécriture typée se présente ainsi comme une généralisation de la réécriture classique.

Théorème 5.4

Une réécriture typée possède la propriété de terminaison finie, si tous ses systèmes de réécriture subordonnés sont nœthériens.

Preuve :

En effet, si tous les systèmes de réécriture subordonnés à une réécriture typée $\mathfrak{T}$ sont nœthériens, alors, lors de la réécriture $t \xrightarrow{\mathfrak{T}} t'$, la réécriture de t_j en t_{j+1} se fait au terme d'un nombre fini d'étapes n_j pour tout $j \in [1..n]$ (où n est le cardinal de l'ensemble F associé à la réécriture $t \xrightarrow{\mathfrak{T}} t'$). La réécriture typée de t en t' s'effectue donc en $\sum_{j=1}^{n} n_j$ étapes. Ce qui montre que la réécriture typée possède la propriété de terminaison fini. ∎

5.8 Arbres – Représentation des termes dans *SURVEYOR*

Lorsqu'on doit écrire un système informatique basé sur la connaissance, il est clair que l'isomorphie entre la représentation interne des objets manipulés et leur représentation externe permet une lucidité indéniable du processus de raisonnement. Ainsi, nous avons choisi d'exprimer les termes dans le système *SURVEYOR* par une représentation relativement complexe puisque nous considérons des arbres et non seulement des chaînes de caractères.

Définition 5.14

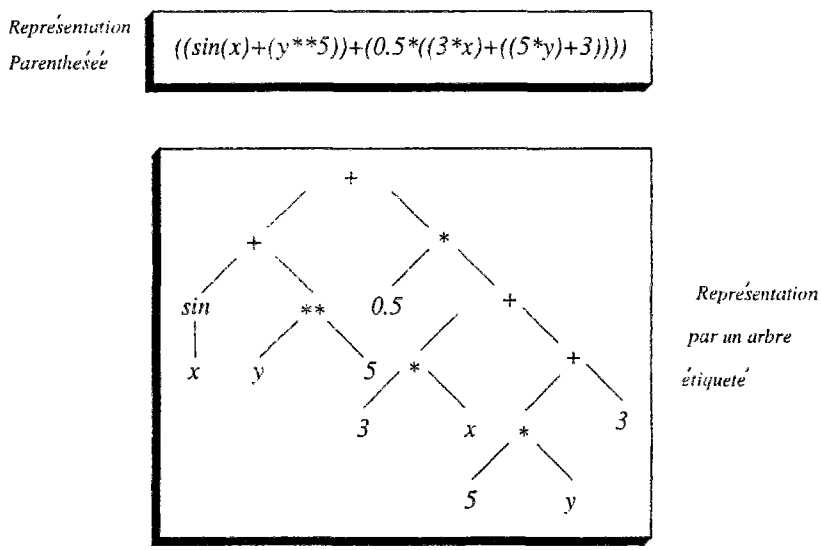
Un arbre est un ensemble fini partiellement ordonné qui vérifie les deux axiomes suivants:

i) Il existe un élément plus petit que tous les autres appelé racine de l'arbre.

ii) Les minorants de tout élément forment un ensemble totalement ordonné.

Donc, les successeurs d'un nœud (i.e. élément de l'arbre) sont des nœuds plus grand. De plus, si l'on munit chaque nœud d'une relation d'ordre totale sur l'ensemble de ses successeurs, on obtient un arbre ordonné[4] [Fig. 5.2].

Le système *SURVEYOR* manipule des relations et des expressions sous forme d'arbres étiquetés. Un arbre étiqueté est constituer de nœuds comportant chacun une étiquette qui désigne une information atomique (i.e. constante). A chaque nœud sont reliés, par un certain nombre de



Les deux représentations ci-dessus de l'expression mathématique $\sin x + y^5 + 0.5\,(3\,x + 5\,y + 3)$ sont équivalentes.

Fig. 5.2: Représentations d'une expression mathématique

"branches", autant de nouveaux arbres que nécessaires. Ainsi, la structure d'arbre peut représenter toute sorte de connaissances. Plusieurs raisons justifient le choix des arbres comme structure de données:

- Leur utilisation est à priori indéterministe.
- Leur capacité de représenter des informations complexes, sous une forme organisée hiérarchiquement.

[4] C'est exactement cette idée que nous avons utilisée le parcours de l'arbre topologique d'un système articulé de façon à pouvoir établir des modèle itératifs [Chapitres 3 et 4].

- La simplicité de leur manipulation informatique.
- La rigueur du formalisme algébrique qui les représente.
- La lisibilité incomparable de leur représentation sous forme graphique.

Malgré tous ces avantages, leur maniement (concaténation, substitution, ...etc) n'est pas toujours facile. C'est pourquoi, dans la pratique, il est très important d'employer une notation parenthésée dans laquelle tout arbre est codé sous forme d'une liste entre parenthèses[5]. La représentation des termes peut se faire alors à l'aide d'expressions parenthésées qui contiennent des variables[6] qui peuvent éventuellement pointer vers d'autres termes qui à leur tour portent sur des variables et ainsi de suite. Un arbre qui contient des variables représente une infinité d'arbres possibles selon les valeurs que l'on donnera à chacune de ses variables. Ainsi, la forme générale de l'arbre est fixée mais son architecture exacte va dépendre des valeurs que vont prendre ses variables. Lorsqu'on fixe les valeurs des variables d'un terme donné, on peut calculer sa valeur en remplaçant chaque occurrence de chacune des variable par sa valeur. Nous appellerons cette l'affectation qui permet une telle transformation syntaxique "affectation forestière" [Fig. 5.3].

5.9 Formalisation de l'affectation forestière

Attirons tout d'abord l'attention du lecteur sur le fait que le remplacement d'un sous-terme d'un terme à une occurrence donnée et l'affectation forestière sont deux aspects différents différents de la substitution. Elle correspondent, en effet, à deux opérations différentes sur les termes. Ainsi, elles seront notés différemment dans la suite de ce chapitre.

Définition 5.15

Soient $t, t_1, t_2, \cdots, t_n$, $n+1$ termes de l'algèbre $T(\Sigma, \vartheta)$ et $x_1, x_2, \cdots, x_n$, n variables distinctes de ϑ. Nous appellerons affectation forestière du terme t pour les valeurs des variables $x_1, x_2, \cdots, x_n$ prises successivement égales à $t_1, t_2, \cdots, t_n$, le terme résultat de la substitution de t_i à chaque occurrence de la variable x_i pour $i \in [1..n]$.

En d'autres termes une affectation forestière t pour les valeurs des variables $x_1, x_2, \cdots, x_n$ prises successivement égales à $t_1, t_2, \cdots, t_n$, est le résultat d'une substitution σ telle que:

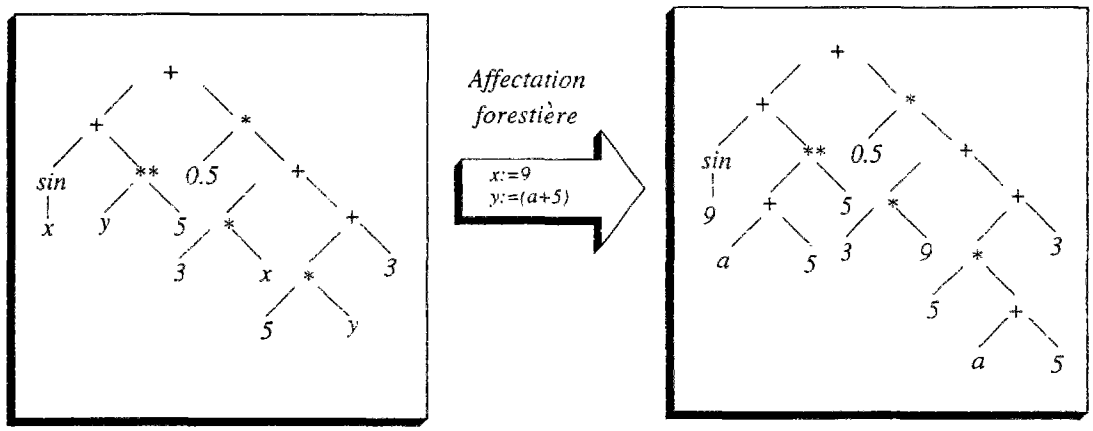
- son domaine de définition est: $dom(\sigma) = \{x_1, x_2, \cdots, x_n\}$,
- σ filtre les variables $x_1, x_2, \cdots, x_n$ vers les termes $t_1, t_2, \cdots, t_n$ respectivement.

Ainsi, le terme résultant d'une telle affectation sera noté dans la suite:

$$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n]$$

[5]C'est l'une des raisons qui ont motivé notre choix du langage Lisp pour l'écriture du système *SURVEYOR*.

[6]C'est à dire des objets dont la valeur n'est pas spécifiée.



Cette figure représente la transformation syntaxique associée à une affectation forestière. Toutes les occurrences des variables x et y sont remplacées par leurs valeurs respectives dans l'affectation forestière. Cette affectation consiste à substituer les occurrences de la variables en "greffant" leurs valeurs, autant de fois que nécessaires, sur l'arbre. Cette "greffe" peut avoir la forme d'un atome ou d'un arbre.

Fig. 5.3: Affectation forestière

Lemme 5.5

Soient $t, t_1, t_2, \cdots, t_n$, $n+1$ termes de l'algèbre $T(\Sigma, \vartheta)$ et $x_1, x_2, \cdots, x_n$ et n variables distinctes de ϑ. Posons $u = t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n]$, alors:

$$V_u = (V_t \setminus \{x_1, x_2, \cdots, x_n\}) \cup \bigcup_{i=1}^{n} V_{t_i}$$

Preuve :

La démonstration découle directement de la définition 5.15 et de la relation (Eq.5.5). ∎

Propriétés 5.6

Soient $t, t_1, t_2, \cdots, t_n$, $n+1$ termes de l'algèbre $T(\Sigma, \vartheta)$ et $x_1, x_2, \cdots, x_n$ et n variables distinctes de ϑ, nous avons les quatre propriétés suivantes:

(P1) si t est un terme clos (i.e. $t \in \Sigma^*$), alors:

$$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = t$$

(P2) si $t = x$ (où x est une variable de $\vartheta \setminus \{x_1, x_2, \cdots, x_n\}$, alors:

$$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = x$$

(P3) s'il existe $i \in [1..n]$, tel que $t = x_i$, alors:

$$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = t_i$$

(P4) si f est un symbole de fonction d'arité $m \geq 1$ et s'il existe $u_1, u_2, \cdots, u_m \in T(\Sigma, \vartheta)$ tels que $t = f(u_1, u_2, \cdots, u_m)$, alors:

$$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = f(v_1, v_2, \cdots, v_m)$$

où $v_i = u_i[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n]$, $\forall i \in [1..m]$.

Preuve :

La démonstration de ces propriété découlent directement de la définition de l'affectation forestière et de la proposition 5.1. ∎

Lemme 5.7

Soient $t, t_1, t_2, \cdots, t_n$, $n+1$ termes de l'algèbre $T(\Sigma, \vartheta)$ et $x_1, x_2, \cdots, x_n$, n variables distinctes de $\vartheta \setminus V_t$ (où V_t représente l'ensemble des variables qui apparaissent sur les nœuds de l'arbre du terme t) : $t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = t$.

Preuve :

Elle découle par induction sur t, à partir des propriétés (P1) et (P2). ∎

Maintenant considérons $\mathfrak{S}_n$, l'ensemble des permutations de l'ensemble $[1..n]$.

Théorème 5.8

Pour toute permutation $\sigma \in \mathfrak{S}_n$, on a:

$$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = [x_{\sigma(1)} \leftarrow t_{\sigma(1)}, x_{\sigma(2)} \leftarrow t_{\sigma(2)}, \cdots, x_{\sigma(n)} \leftarrow t_{\sigma(n)}]$$

Preuve :

La démonstration de ce théorème peut être faite en composant une récurrence sur la longueur du terme t par une induction sur t. En effet:

- si $|t| = 1$, alors, trois cas:
  (a) t est un symbole de constante, donc:

  $$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = t = t[x_{\sigma(1)} \leftarrow t_{\sigma(1)}, x_{\sigma(2)} \leftarrow t_{\sigma(2)}, \cdots, x_{\sigma(n)} \leftarrow t_{\sigma(n)}]$$

  compte tenu de la propriété (P1).

  (b) $t = x$ est une variable distincte de $x_1, x_2, \cdots, x_n$, alors:

  $$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = x = t[x_{\sigma(1)} \leftarrow t_{\sigma(1)}, x_{\sigma(2)} \leftarrow t_{\sigma(2)}, \cdots, x_{\sigma(n)} \leftarrow t_{\sigma(n)}]$$

  compte tenu de la propriété (P2).

(c) il existe $i \in [1..n]$, tel que, $t = x_i$, alors:

$$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = x_i = t[x_{\sigma(1)} \leftarrow t_{\sigma(1)}, x_{\sigma(2)} \leftarrow t_{\sigma(2)}, \cdots, x_{\sigma(n)} \leftarrow t_{\sigma(n)}]$$

compte tenu de la propriété (P3).

- supposons maintenant que le théorème est vérifié pour toutes les valeurs de $|t|$ inférieures ou égales à m ($1 \leq |t| \leq m$) et montrons qu'il le reste pour $|t| = m + 1$. Donc il existe un symbole de fonction f et m termes $u_1, u_2, \cdots, u_m \in T(\Sigma, \vartheta)$ tels que:

$$t = f(u_1, u_2, \cdots, u_m)$$

donc, d'après la propriété (P4), on a:

$$t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] = f(v_1, v_2, \cdots, v_m)$$

où $v_i = u_i[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n]$, $\forall i \in [1..m]$.
Or, $\forall i \in [1..m] : 1 \leq |u_i| \leq |t| - 1 = m$, donc d'après l'hypothèse de récurrence:

$$\forall i \in [1..m], \forall \sigma \in \mathfrak{S}_n : \; v_i = v_i^\sigma \text{ où } v_i^\sigma = u_i[x_{\sigma(1)} \leftarrow t_{\sigma(1)}, x_{\sigma(2)} \leftarrow t_{\sigma(2)}, \cdots, x_{\sigma(n)} \leftarrow t_{\sigma(n)}]$$

i.e. compte tenu de la propriété (P4):

$$\begin{aligned} t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n] &= f(v_1^\sigma, v_2^\sigma, \cdots v_m^\sigma) \\ &= t[x_{\sigma(1)} \leftarrow t_{\sigma(1)}, x_{\sigma(2)} \leftarrow t_{\sigma(2)}, \cdots, x_{\sigma(n)} \leftarrow t_{\sigma(n)}] \end{aligned}$$

C.Q.F.D. ∎

Définition 5.16

Soient $t, t_1, t_2, \cdots, t_n$, $n+1$ termes de $T(\Sigma, \vartheta)$ et $x_1, x_2, \cdots, x_n$, $y_1, y_2, \cdots, y_m$, $m+n$ variables distinctes de ϑ. Nous dirons que les variables $y_1, y_2, \cdots, y_m$ sont libres dans l'affectation forestière $u = t[x_1 \leftarrow t_1, x_2 \leftarrow t_2, \cdots, x_n \leftarrow t_n]$, si les variables $y_1, y_2, \cdots, y_m$ ne figurent pas parmi les variables introduites par cette affectation. i.e.:

$$\{y_1, y_2, \cdots, y_m\} \cap \bigcup_{i=1}^{n} V_{t_i} = \emptyset \qquad \text{(Eq.5.6)}$$

Théorème 5.9

Soient $t, u_1, u_2, \cdots, u_n, v_1, v_2, \cdots, v_m$, $m+n+1$ termes de l'algèbre $T(\Sigma, \vartheta)$ et $x_1, x_2, \cdots, x_n$, $y_1, y_2, \cdots, y_m$, $m+n$ variables distinctes de ϑ. Si les variables $y_1, y_2, \cdots, y_m$ sont libres dans l'affectation forestière $u = t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n]$, alors:

$$\begin{aligned} t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n][y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m] = \\ t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n, y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m] \end{aligned}$$

Preuve :

Procédons encore par induction sur t:

- Si $|t| = 1$, alors, il faut envisager 3 cas possibles:

(a) si t est un symbole de constante ou une variable distincte de $x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_m$, d'après le lemme 5.7:

$$t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n][y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m] = t = t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n, y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m]$$

(b) s'il existe $i \in [1..n]$, tel que, $t = x_i$, alors compte tenu du fait que $y_1, y_2, \cdots, y_m$ soient des variables libres dans l'affectation forestière $t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n]$:

$$t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n][y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m] = u_i[y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m] = u_i = t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n, y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m]$$

(c) s'il existe $i \in [1..m]$, tel que, $t = y_i$, comme les variables $x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_m$ sont distinctes et compte tenu du lemme 5.7:

$$t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n][y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m] = t[y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m] = v_i = t[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n, y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m]$$

- nous supposerons par récurrence que le théorème est vérifié pour tout terme t de longueur inférieure ou égale à l (i.e. $1 \leq |t| \leq l$) et montrons qu'il le reste à l'ordre $|t| = l + 1$. En effet, comme $|t| > 1$, il existe un symbole de fonction f et s termes $w_1, w_2, \cdots, w_s$ tels que: $t = f(u_1, u_2, \cdots, u_m)$ avec $1 \leq |w_i| \leq l$ pour $i \in [1..s]$. Ainsi, d'après l'hypothèse de récurrence, pour tout $i \in [1..s]$:

$$w_i[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n][y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m] = w_i[x_1 \leftarrow u_1, x_2 \leftarrow u_2, \cdots, x_n \leftarrow u_n, y_1 \leftarrow v_1, y_2 \leftarrow v_2, \cdots, y_m \leftarrow v_m]$$

ce qui achève la démonstration. ∎

Ainsi, la réalisation d'un système d'intelligence artificielle dans cette représentation suppose nécessairement:

- Une implémentation particulière des principes généraux du traitement des arbres.
- Un contrôle particulier du processus de réécriture, tel qu'à tout instant, le choix de la règle de réécriture (à "armer") soit en fonction de l'état actuelle de la réduction du terme.

5.10 Règles de production

Les règles de production sont des parcelles de connaissances de la forme:

Si *condition* **Alors** *action*

C'est à dire si les prémisses (i e. *condition*) sont vérifiées alors une action (i.e. *action*) peut être déclenchée. Ainsi exprimée, la connaissance est représentée de manière purement déclarative et par la suite: la connaissance d'une part et le système de contrôle qui permet son exploitation d'autre part peuvent être totalement indépendants (ce qui n'est pas le cas dans les descriptions algorithmiques traditionnelles). Au cours de l'exécution, une règle de production dont la partie *condition* est satisfaite peut être activée par l'interpréteur *SURVEYOR*.

5.11 Architecture du système *SURVEYOR*

Comme tout système à base de règles, le système *SURVEYOR* se compose de trois parties essentielles:

- La base des faits.
- La base de connaissance.
- Le moteur d'inférence.

Nous nous proposons maintenant de décrire l'architecture du système *SURVEYOR*, son modèle de représentation des connaissances et son mode de contrôle de cette connaissance.

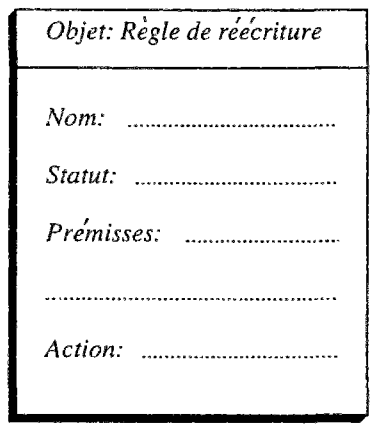
5.11.1 Base des faits

La base des faits du système *SURVEYOR* est une liste (représentation Lisp d'un arbre étiqueté) qui représente le terme en cours de réduction. Il s'agit d'une mémoire de travail caractérisée par un arbre qui s'enrichit de faits nouveaux (i.e. transformations syntaxiques) découverts par le mécanisme de raisonnement jusqu'à ce que celui-ci parvienne à une solution. Lorsque le mécanisme de raisonnement parvient à un terme irréductible la base de fait est remise à zéro (i.e. liste vide).

5.11.2 Base de connaissance

La base de connaissance présente la mémoire à long terme d'un système d'intelligence artificielle. Dans le cas du système *SURVEYOR*, elle décrit le savoir faire opératoire des règles de réécriture et elle se compose essentiellement:

- D'un ensemble de systèmes de réécriture associés à des statuts de transformation différents. Chaque système de réécriture se compose d'un nombre de schémas de règles de réécriture codées sous forme de règles de production structurées. C'est à dire, chacune des règles est représentée par un ensemble de champs [Fig. 5.4]:
  - Le nom de la règle: il permet au système de contrôle de reconnaître une règle parmi les autres.
  - Le statut de transformation: spécifiant le contexte général qui nécessite l'activation de cette règle. Conformemment à la partie théorique de cet exposé, il faut noter que lorsqu'il s'agit d'une règle de réécriture le statut de transformation n'est autre que l'attribut de réécriture associé au système de réécriture (subordonné à la réécriture typée) dans lequel elle est considérée. La technique d'attribution d'un statut de transformation aux règles va permettre au système de contrôle de *SURVEYOR* une meilleure efficacité de la résolution de conflits. En effet, cela va lui permettre de limiter le nombre de tentatives infructueuses au cours de chaque cycle d'inférence. En outre, il permet de prendre en compte les connaissances des concepts profonds (réécriture typée et connaissances heuristiques) sur lesquels se base le raisonnement et d'éviter au système la



Une règle de réécriture se présente sous la forme d'un objet structuré dans le système *SURVEYOR*. Chaque règle de réécriture est codée par un ensemble de quatre champs: Nom, Statut, Prémisses et Action

Fig. 5.4: L'objet: Règle de réécriture

non-terminaison du processus de réécriture. En effet, des règles de réécritures susceptibles d'amener le processus de réécriture à diverger peuvent être classées avec des attributs de réécriture différents. La propriété de terminaison fini est alors assurée sous les hypothèses du théorème 5.4.

Exemple

Des règles inverses telles que:

$$\begin{array}{lcl} R_1 & : & f(x_1, \cdots, x_n) \longrightarrow g(x_1, \cdots, x_n) \\ R_2 & : & g(x_1, \cdots, x_n) \longrightarrow f(x_1, \cdots, x_n) \end{array}$$

peuvent être utilisées pour la réécriture d'un même terme avec des attributs différents A_1 et A_2. Bien entendu, à ces deux attributs peuvent être associées d'autres règles tant que les hypothèses du théorème 5.4 sont vérifiées.

- Une partie prémisses: traduisant la ou les conditions élémentaires immédiates qui permettent le déclenchement de la règle. Ces conditions sont représentées par un prédicat qui permet de tester si le membre gauche de la règle de réécriture s'identifie à un terme de la base des faits. Cette méthode de représentation des règles va permettre non seule-

ment de représenter des règles de réécriture simples, mais aussi des règles de réécriture conditionnelles. C'est à dire des règles de la forme:

$$\textbf{Si } cond(V_g) \textbf{ Alors } g \longrightarrow d$$

où *coud* est un prédicat qui porte sur l'ensemble des variables V_g du membre gauche g de la règle $g \longrightarrow d$.

Exemple

Cela permet de prendre en compte des règles de la forme:

R_1	:	si $x = 0$ alors $f(x) \longrightarrow g(x)$
R_2	:	si $x \neq 0$ alors $f(x) \longrightarrow h(x)$

- Une partie action: qui permet de transformer syntaxiquement l'arbre identifié au membre gauche de la règle de réécriture. Cette transformation est décrite par le membre droit de la règle en substituant les occurrences de chaque variable par la valeur issue de l'appariement (anglais: patern matching) des variables du membre gauche de la règle de réécriture au sous-termes de l'expression mathématique en question.

- Des règles qui permettent de passer d'un système de réécriture à un autre. Ces règles sont en fait des méta-règles, puisqu'elles portent sur des heuristiques qui traduisent une méta-connaissance.

5.11.3 Codage interne des règles

A titre d'exemple, la règle de réécriture $(x_1 + \varepsilon * x_2) + (y_1 + \varepsilon * y_2) \longrightarrow (x_1 + y_1) + \varepsilon * (x_2 + y_2)$, qui permet d'opérer syntaxiquement la somme de deux nombres duaux, peut être codée dans le système *SURVEYOR* sous différentes formes:

- **Implémentation semi-déclarative**: ce premier mode d'implémentation permet déclarer certains champs de l'objet *règle de réécriture* de façon procédurale. L'appellation *semi-déclarative* signifie ici que la règle est déclarée sous forme de champs suivant un prototype bien défini:

Nom:	`<x1+eps*x2>+<y1+eps*y2> --> <x1+y1>+eps*<x2+y2>`
statut:	`dual_number`
Prémisses:	`(add_dual_form left)`
Action:	`'(+ (+ ,x1 ,y1) (* eps (+ ,x2 ,y2)))`

où `left` désigne le terme gauche de la règle de réécriture et la fonction `add_dual_form` est définie par:

```
(de add_dual_form (form)
      (and (not (atom form))
           (match '(+ (? xx) (? yy)) form)
           (dual_form xx x)
           (dual_form yy y)
      )
)
```

Cette définition se base sur la fonction d'appariement `match` (du système de contrôle) et la fonction `dual_form` qui cherche à déterminer un renommage dual de l'expression à laquelle elle est appliquée. Examinons, à présent, l'implémentation de la fonction `dual_form` qui est propre au codage de cette règle de réécriture:

```
(de dual_form (var root_var)
      (and (not (atom var))
           (match '(+ (? ,(make_name root_var 1))
                      (* eps (? ,(make_name root_var 2)))
                   )
                   var
           )
      )
)
```

où `make_name` est un constructeur de nom de variables défini par:

```
(de make_name (root num)
      (implodech '(,root ,num))
      )
)
```

Le constructeur `make_name` permet de construire de nouveaux noms de variables à partir d'une racine `root` (chaîne de caractères) et un nombre entier `num`. Ainsi, à titre d'exemple, à l'exécution sous le top level du dialecte le_lisp:

```
? (make_name 'x 1)
= x1
```

Cette façon de coder les règles est relativement complexe, mais, elle présente un intérêt essentiel qui est celui de présenter les prémisses sous forme d'un prédicat. Ainsi, toute sortes d'heuristiques. relatives à l'application d'une règle, peuvent être décrites dans ce prédicat.

- **Implémentation déclarative**: le système *SURVEYOR* offre la possibilité de coder une règle de réécriture sous une forme simple et relativement naturelle:

Nom:	`<x1+eps*x2>+<y1+eps*y2> --> <x1+y1>+eps*<x2+y2>`
statut:	`dual_number`
Prémisses:	`(+ (+ (? x1) (* eps (? x2))) (+ (? y1) (* eps (? y2))))`
Action:	`'(+ (+ ,x1 ,y1) (* eps (+ ,x2 ,y2)))`

Ainsi, la règle de réécriture est codée sous une forme déclarative. C'est l'interprète du système *SURVEYOR* qui va se charger de l'appariement de la partie prémisses qui sous cette forme n'est plus un prédicat mais une expression mathématique qui contient quatre variables notées respectivement `(? x1)`, `(? x2)`, `(? y1)` et `(? y2)`.

Par ailleurs, le système *SURVEYOR* offre la possibilité d'appliquer la règle:

```
<x1+eps*y1>+<x2+eps*y2> --> <x1+x2>+eps*<y1+y2>
```

dans des contextes différents. Ainsi, à titre d'exemple, cette règle peut être appliquée avec les statuts `dual_number` et `dual_vector`. Il suffit, dans ce cas d'exprimer la règle avec la suite des statuts de réécriture. Cette approche permet d'attribuer un sens sémantique à l'application des règles de réécriture.

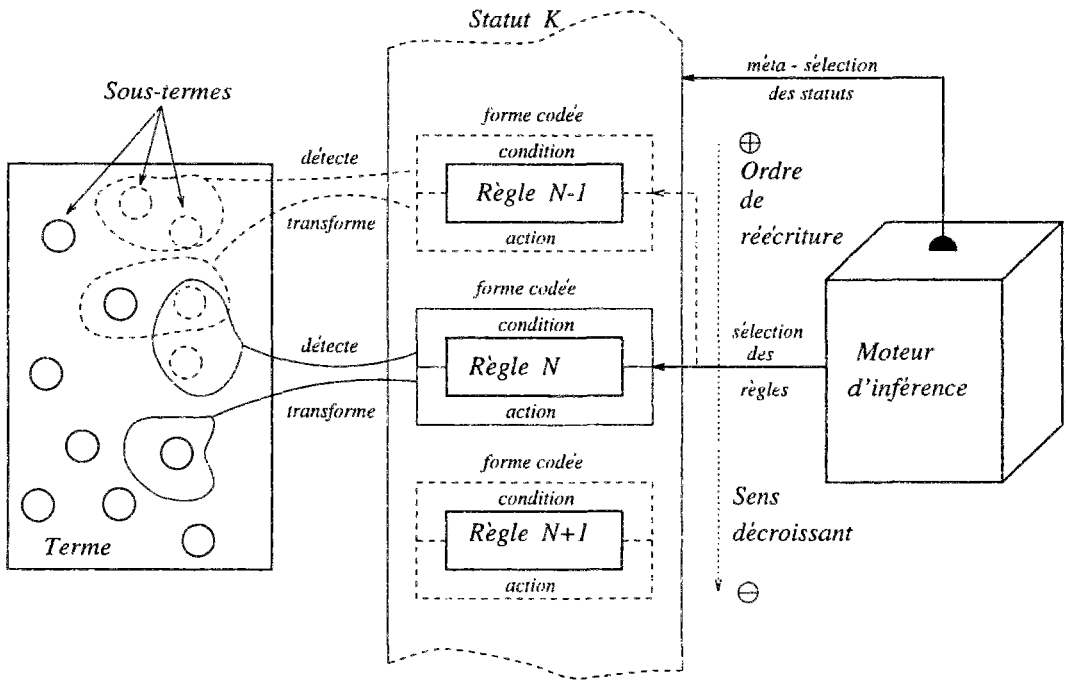
Enfin, c'est les méta-règles qui vont permettre d'exprimer la méta-connaissance décrivant le passage d'un statut de transformation à un autre. Ainsi, la méta-règle permettant le passage du statut `dual_number` au statut `dual_vector` va s'exprimer sous la forme:

Nom:	`dual_number ==> dual_vector`
statut:	`dual_number`
Prémisses:	`(($ path))`
Action:	`(prog () (setq current_goal 'dual_vector) (return path) )`

On voit bien sur cette exemple qu'une méta-règle présente les mêmes champs qu'une règle sauf qu'elle décrit une méta-connaissance (ici c'est le passage du statut `dual_number` au statut `dual_vector`).

5.11.4 Moteur d'inférence pour la réécriture typée

Lorsque nous voulons déduire d'un ensemble d'axiomes équationnels quelque propriété ou encore transformer une expression dans le contexte définis par ces axiomes, nous parcourons d'un "mouvement de pensée" continue l'ensemble de ces équations jusqu'à ce que nous ne pourrons plus en appliquer aucune. C'est ainsi que nous exploitons la connaissance computationnelle relative



Une fois un statut de transformation sélectionné, les règles d'inférence répertoriées sous ce statut peuvent contribuer à la transformation syntaxique d'un terme. Lorsque le système de contrôle identifie une sous-expression du terme au membre gauche d'une règle de réécriture, il déclenche la transformation décrite par cette règle. Pour ce faire, le moteur d'inférence du système *SURVEYOR* dispose d'un sélectionneur et d'un méta-sélectionneur:

- le méta-sélectionneur spécifie tour-à-tour les statuts de transformation possibles.
- le sélectionneur spécifie si une règle examinée est déclenchable ou non.

Fig. 5.5: Modèle conceptuel de *SURVEYOR*

à une théorie axiomatique. Le résultats final est atteint à travers un ensemble d'étapes intermédiaires par un enchaînement de raisonnements. Cette faculté d'avoir en soi un procédé déductif permettant de rendre raison d'une connaissance computationnelle peut être mécanisée.

Le moteur d'inférence du système *SURVEYOR* est un programme chargé d'exploiter les connaissances et les faits pour mener un raisonnement qui intègre plusieurs systèmes de réécriture. Le moteur d'inférence du système *SURVEYOR* raisonne sur des variables substituables tour à tour par des termes. Une telle représentation des connaissances qui autorise la présence de variables est dite d'ordre 1. Cette représentation confère au système une grande puissance d'expression où chaque règle représente un schéma qui résume un ensemble de règles d'ordre 0 (i.e. où n'interviennent que

des constantes). Le mécanisme de raisonnement de *SURVEYOR* est un mécanisme de chaînage mixte: un interprète à chaînage avant pour faire de la réécriture dans un contexte défini par un statut donné, guidé par un méta-interprète à chaînage arrière permettant de changer le contexte de réécriture. Il consiste, une fois le statut de transformation spécifié, à produire des règles de la forme:

Si *condition* **Alors** *action*

A chaque étape du raisonnement, le moteur d'inférence parcours trois phases:

(a) Restriction du domaine: par détection des règles dont la partie prémisses est vérifiée (vérification des spécifications conditionnelles). Notons que les règles sont disposées en vrac dans la base de règles. C'est le système de contrôle qui, en fonction de l'état de réduction du terme, va décider quelles règles peuvent être déclenchées. A ce stade les règles sont classées en deux catégories de règles:

- les *règles déclenchables*: sont les règles susceptibles d'être activées à un moment ou un autre.
- les *règles inactives*: sont les règles qui ne peuvent être déclenchées.

(b) Résolution de conflits: sélection d'une règle parmi les candidates. En effet, lorsque plusieurs règles sont candidates à être déclenchées, elles le sont à priori à égalité de chance. Pour "mener à bien" la réécriture (i.e. convergence de la réécriture) , *SURVEYOR* offre la possibilité de décrire la priorité des règles dans une pile de priorité où les règles sont repérées par leurs noms. Cet ordre de priorité n'est en fait autre que l'ordre dans lequel le système de réécriture va "fonctionner". C'est cette heuristique qui permettra au système *SURVEYOR* de savoir le moment venu quelle règle il doit choisir parmi les règles candidates.

(c) Déclenchement de la règle sélectionnée: la structure du graphe représentant le terme en cours va être modifiée compte tenu de la structure syntaxique du membre de droite de la règle déclenchée.

Mais, insistons sur le fait qu'une règle qui se trouve dans une pile à examiner de manière prioritaire n'est pas nécessairement déclenchable pour autant et que les priorités de déclenchement des règles sont des priorités locales qui permettent par cette heuristique de classer les règles correspondant à un même statut de réduction les unes par rapport aux autres.

Par ailleurs, le système de contrôle de *SURVEYOR* est basé sur un contrôle de la profondeur de développement. En effet, on peut avoir besoin de déclencher une même règle plusieurs fois. Ainsi, le cycle va reprendre, autant de fois que nécessaire, jusqu'à ce que le terme devienne irréductible et atteint une forme normale. En d'autres termes, jusqu'à ce qu'il ne serait plus possible de déclencher aucune règle de la base de connaissance. Ainsi, lorsqu'une règle a été déclenchée avec un jeu d'instanciations, le moteur d'inférence va tenter de la redéclencher avec d'autres jeux

d'instanciations jusqu'à ce qu'il ne puisse plus en trouver de nouveaux. Mais une fois déclenchée avec toutes les instanciations possibles, la règle est désactivée. Elle ne sera réactivée que si des sous-termes correspondant à ses prémisses réapparaissent dans le terme en cours de réduction.

Pour le contrôle de la réécriture, *SURVEYOR* offre la possibilité de décrire la connaissance mathématique par un jeu de systèmes de réécriture. A chaque système de réécriture, comme on l'a déjà vu, est associé un statut de transformation par réécriture. Ainsi, on peut décrire des connaissances heuristiques qui permettent à *SURVEYOR* de savoir dans quelles conditions il est intéressant de chercher à appliquer les règles de tel ou tel système de réécriture. Cette méthode permet au système de limiter le nombre de tentatives infructueuses au cours de la recherche d'une solution.

Le système *SURVEYOR* va résoudre un problème mathématique "bien posé" par réécriture jusqu'à saturation de tous les paquets de règles d'inférence. Chaque paquet contient alors un ensemble de règles de réécriture et une méta-règle qui permet de pointer vers un nouveau statut de réécriture.

5.11.5 Exemple

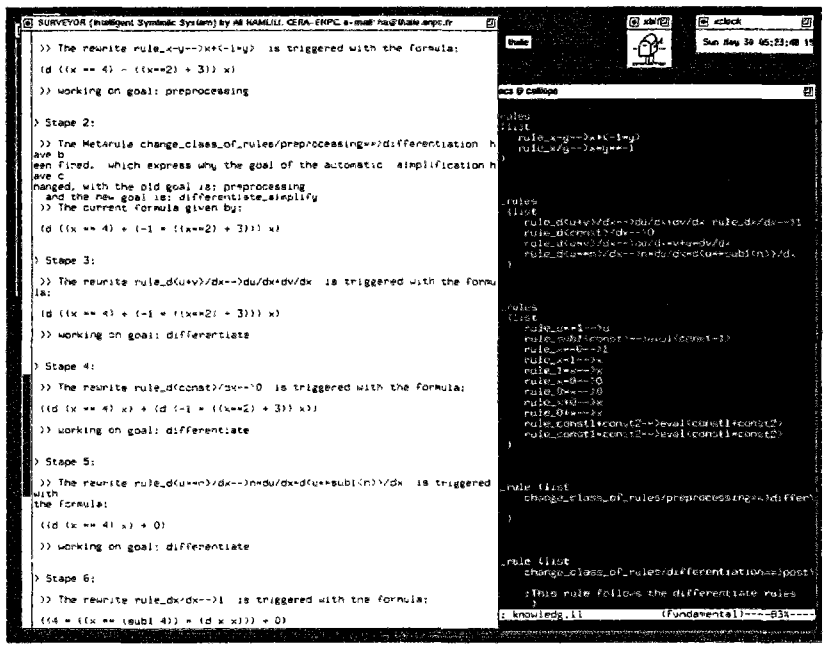
Une méthode efficace pour la description déclarative du problème classique de la différentiation formelle des polynômes consiste en la donnée des règles sous forme de trois systèmes de réécriture:

- Le système de réécriture pré-processeur:

Règle *pré-processeur*$_1$:	$x - y \longrightarrow x + (-1 * y)$
Règle *pré-processeur*$_2$:	$\frac{x}{y} \longrightarrow x * y^{-1}$

pour éviter de programmer les règles de réécriture traduisant la dérivation des différences et des quotients, qui peuvent être déduites des règles de réécriture sur les sommes et les produits.

- Le système de réécriture de différentiation:



Dans l'exemple ci-contre, le système *SURVEYOR* procède à la différentiation symbolique d'une fonction polynomiale.

Fig. 5.6: Session de travail sur *SURVEYOR*

Règle *différentiation*$_1$:	$\frac{d(const)}{dx} \longrightarrow 0$
Règle *différentiation*$_2$:	$\frac{dx}{dx} \longrightarrow 1$
Règle *différentiation*$_3$:	$\frac{d(u+v)}{dx} \longrightarrow \frac{du}{dx} + \frac{dv}{dx}$
Règle *différentiation*$_4$:	$\frac{d(u*v)}{dx} \longrightarrow \frac{du}{dx} * v + u * \frac{dv}{dx}$
Règle *différentiation*$_5$:	$\frac{d(u^n)}{dx} \longrightarrow n * (u^{n-1}) * \frac{du}{dx}$

qui traduisent un système minimal de règles permettant de décrire la différentiation des fonctions polynomiales.

- Le système de réécriture post-processeur:

Règle *post-processeur*$_1$:	$x^1 \longrightarrow x$
Règle *post-processeur*$_2$:	$x^0 \longrightarrow 1$
Règle *post-processeur*$_3$:	$x * 1 \longrightarrow x$
Règle *post-processeur*$_4$:	$1 * x \longrightarrow x$
Règle *post-processeur*$_5$:	$x * 0 \longrightarrow 0$
Règle *post-processeur*$_6$:	$0 * x \longrightarrow 0$
Règle *post-processeur*$_7$:	$x + 0 \longrightarrow x$
Règle *post-processeur*$_8$:	$0 + x \longrightarrow x$
Règle *post-processeur*$_9$:	$x + (-1 * y) \longrightarrow x - y$
Règle *post-processeur*$_{10}$:	$x * y^{-1} \longrightarrow \frac{x}{y}$

pour permettre de simplifier les expressions obtenue par les règles de dérivation.

- Pour obtenir des arbres qui ont des représentations semblables, les propriétés de commutativité et d'associativité peuvent être exploitées à cette fin en ajouttant les règles orientées suivantes:

Règle *équivalent*$_1$:	$const + x \longrightarrow x + const$
Règle *équivalent*$_2$:	$(x + y) + z \longrightarrow x + (y + z)$
Règle *équivalent*$_3$:	$x * const \longrightarrow const * x$
Règle *équivalent*$_4$:	$x * (y * z) \longrightarrow (x * y) * z$

Ces quatre règles permettent ainsi de normaliser la représentation des expressions en écrivant le facteur constant à "gauche" dans un produit et le terme constant d'une somme à "droite" des l'expressions mathématiques.

Ainsi, à ces quatre systèmes de réécriture peuvent être attribués trois statuts de réécriture différents. Bien entendu, les trois systèmes de réécriture peuvent être enrichis de règles de façon à ce que le processus de différentiation soit applicable à des fonctions plus générales.

A titre d'exemples:

1) la règle de réécriture "*différentiation*$_3$" traduit en faite une règle de réécriture exprimée dans un spécification conditionnelle:

$$\textit{différentiation}_3 \equiv \textbf{Si } u \textit{ est constante } \textbf{Alors: } \frac{du}{dx} \longrightarrow 0$$

"*u est constante*" tient lieu de prédicat de premier ordre portant sur la variable u.

2) la règle de réécriture "*équivalent*$_2$" à pour effet de présenter des formes normales parenthésées telles que toute sous-expression de la forme $(x+y)+z$ soit réécrite sous la forme parenthésée $x+(y+z)$. Cette règle permet en contribution avec la règle "*équivalent*$_1$" de rassembler entre eux les termes numériques. Des schémas de règles de la forme:

$$const_1 + const_2 \longrightarrow evaluate(const_1 + const_2)$$

tiennent le rôle d'évaluateurs.

5.12 Observations

La modélisation par des règles de réécriture offre ainsi un outil conceptuel précis pour représenter une connaissance équationnelle, mais assez général pour ne pas spécifier la manière d'exploiter cette connaissance (description non-algorithmique). Ainsi, on peut remarquer que:

- La connaissance est organisée sous forme de modules indépendants.
- Les différents modules peuvent se traduire sous le format:

  Si *condition* **Alors** *action*

  - La partie condition examine la forme syntaxique du terme en cours de réduction et instancie le cas échéant les variables du membre gauche (partie prémisses de la représentation sous forme de règle de production) de la règle de réécriture candidate. La règle de production est ainsi "armée".
  - Quand la partie prémisses de la règle est instanciée, les variables du membre gauche peuvent être évaluées pour les valeurs de l'affectation forestière associée. Cette règle peut alors contribuer à la réduction du terme et sa partie action est déclenchée.
  - Le sous-terme identifié au membre gauche de la règle subit la transformation syntaxique décrite par le membre droit de la règle de réécriture en instance. L'instanciation du membre droit par l'affectation forestière consiste alors à remplacer toutes les occurrences initiales des variables par les instances correspondantes à l'appariement des variables du membre gauche.
- Chaque règle fonctionne de façon indépendante et elle est déclenchée dès que les variables de son membre gauche peuvent être instanciées.
- La solution est construite de façon incrémentale.
- La stratégie de réduction consiste alors à passer au but de réduction suivant dès que l'espace de recherche du but de réduction en cours est saturé.

- Lorsque tous les statuts de réductions sont saturés, le système rend la forme irréductible du terme au moyen des systèmes de réécriture associés à ces statuts et pour l'ordre de réécriture décrit par la classification de priorité des règles dans leurs systèmes de réécriture.

5.13 Conclusion

Le système *SURVEYOR* s'avère un puissant outil de transformation syntaxique des expressions mathématiques dans des spécifications équationnelles (i.e. réécriture typée). Le prix à payer dans un tel système, lorsque la propriété de Church-Rosser n'est pas vérifiée, est le fait qu'il ne va pas donner toutes les solutions possibles au problème. Cependant, il va donner la solution associée à l'ordre de réécriture décrit par l'ordre de priorité de l'application des règles. La représentation irréductible va différer d'un ordre de description à un autre (selon l'heuristique définissant la priorité de déclenchement des règles). Ainsi, en l'absence d'une heuristique décrivant l'ordre de priorité de déclenchement des règles, on ne peut assurer la terminaison ni la convergence de la réécriture vers une représentation optimale de la solution, mais, à une solution tout simplement. L'obstacle essentiel qui interdit de trouver automatiquement une représentation optimale du problème de réécriture étant celui de "l'explosion combinatoire" des possibilités d'ordonner les systèmes de réécriture proposés pour la résolution du problème considéré. Ce nombre de possibilités croît de manière exponentielle. Actuellement, cette combinatoire est un handicap auquel ne peuvent échapper totalement les applications en intelligence artificielle.

Ainsi, un usage judicieux de la fragmentation de la connaissance computationnelle est fortement recommandé pour augmenter l'efficacité d'emploi de la réécriture typée et tirer un meilleur profit de l'usage des techniques d'intelligence artificielle. Cela permettra, entre autres, d'échapper aux problèmes de l'explosion combinatoire et de diminuer substantiellement le temps de réduction d'une expression.

Enfin, la puissance d'expression de la réécriture typée permet au mathématicien (non informaticien) de réaliser, dans un système tel que *SURVEYOR*, de manière tout à fait naturelle des calculs algébriques décrits en grandes partie par des considérations théoriques et mathématiques et non plus par un codage algorithmique informatique (et par définition non mathématique).

Chapitre 6

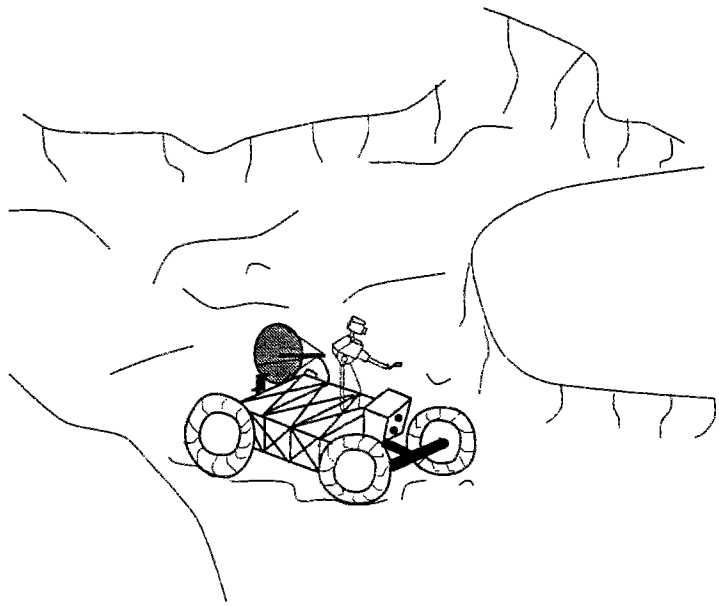
LE SYSTEME SYMBOLIQUE: *MEDUSA MF 77*

6.1 Introduction

La génération automatique des équations de la dynamique des systèmes articulés est, avant tout comme son nom l'indique, l'automatisation du processus d'obtention du code décrivant le comportement dynamique de tels systèmes. En vu d'automatiser l'obtention des équations du mouvement, les premières études eurent pour but la génération de programmes en code Fortran pour la résolution de différents problèmes liés au comportement des systèmes articulés. Rapidement plusieurs logiciels furent réalisés [R. Sturges[STUR-73], W. Khalil [KHAL-78], J.Y.S. Luh [LUH—81], M.G. Aldon [ALDO-82]]. Plus tard, la fascination des ingénieurs et chercheurs (en dynamique des systèmes articulés) par les logiciels de calcul formel, les a amené à développer un nombre d'outils sur la base de ces systèmes informatiques [J. Wittenburg & U. Woltz [WITT-85], Ch. Garnier [GARN-90]]. En effet, la représentation mathématique du comportement dynamique des systèmes articulés, nous le savons tous, ne se réduit pas à la manipulation des données numériques. Avant tout, elle cherche à étudier les propriétés qualitatives du mouvement de tels systèmes. L'utilisation des logiciels de calcul formel a introduit alors de nouvelles options pour le traitement des équations du mouvement:

- Exploitation numérique des expressions formelles pour des valeurs numériques des paramètres symboliques du modèle [Annexe B].
- Exploitation symbolique des expressions formelles [Annexes B et C]:
  - transformation des expressions avec moindres efforts de programmation [Chapitre 7],
  - simplifications et développements formels de différentes sortes: trigonométriques, polynomiaux, algébriques, ...etc [Annexe C],
  - dérivation formelle, intégration formelle, linéarisation, ...etc,
  - génération automatique de codes symboliques optimisés: Fortran ou C [Annexe C].

Mais à coté de ces avantages, l'utilisation classique des logiciels de calcul formels comporte



Pour explorer Mars (projet Mars 2000), le groupement robotique d'intervention sur cite planétaire (regroupant le CNRS, le CEA, l'INRIA et l'ONERA) développent un robot intelligent sur quatre roues motrices. Puisqu'il faut quarante minutes aux ondes électro-magnétiques pour faire l'aller-retour (Terre–Mars) le robot devra être capable de prendre seul des décisions. Ce robot sera équipé d'une camera laser, il devra reconstituer le relief par mesures multiples. Des capteurs internes (gyromètre, accéléromètre) et tactiles contrôleront le robot [Le Journal du CNRS - N° 30 (Juin 1992)].

Fig. 6.1: Robot intelligent autonome dans un environnement variable

certains inconvénients et se fait en général au détriment d'autres formes pratiques de la représentation des équations du mouvement. En effet, un grand nombre d'applications robotiques nécessitent une structuration itérative du modèle:

- applications où l'on souhaite modifier à la main certaines parties des programmes obtenus (sans toucher aux programmes de génération du code) afin de les réutiliser. En effet, la réutilisation de programmes est une pratique de plus en plus fréquentée. Un système principal permet d'obtenir le code source selon un modèle général et les cas particuliers sont obtenus en modifiant à la main certain module de ce code. C'est le cas lorsqu'on souhaite ajouter de certains termes supplétifs obtenus par des procédés autres que celui du logiciel de génération du code. Par exemples: les termes de frottements, changements des conditions limites (i.e.

initialisation des récurrences) pour prendre en compte des efforts supplémentaires exercés sur les organes terminaux (voir même sur tous les membres du robot), ... etc.

- applications robotiques non-manufacturières où l'environnement du robot est mal connu ou sujet à des modifications imprévisibles et où l'on souhaite avoir une commande en temps réel. Le robot doit avoir des réflexes et prendre des décisions en temps réel [Fig. 6.1]. C'est, par exemple, le cas dans les applications spatiales, sous-marines, nucléaires. ... etc, actuellement assurées par télé-opération [B. Espiau [ESPI-86]].

Ainsi, nous avons conçu et réalisé le système *MEDUSA MF77* -provient de l'acronyme de "**ME**echanism **D**ynamics by **U**sing **S**ymbolic **A**lgebra systems"- pour répondre à un besoin double:

- la génération d'un code source itératif optimisé en nombre d'opérations élémentaires (multiplications, additions et évaluations des fonctions trigonométriques) destiné à être intégré dans des programmes de calcul,
- la génération des équations sous forme symbolique, pour faire éventuellement l'objet de différentes transformations formelles.

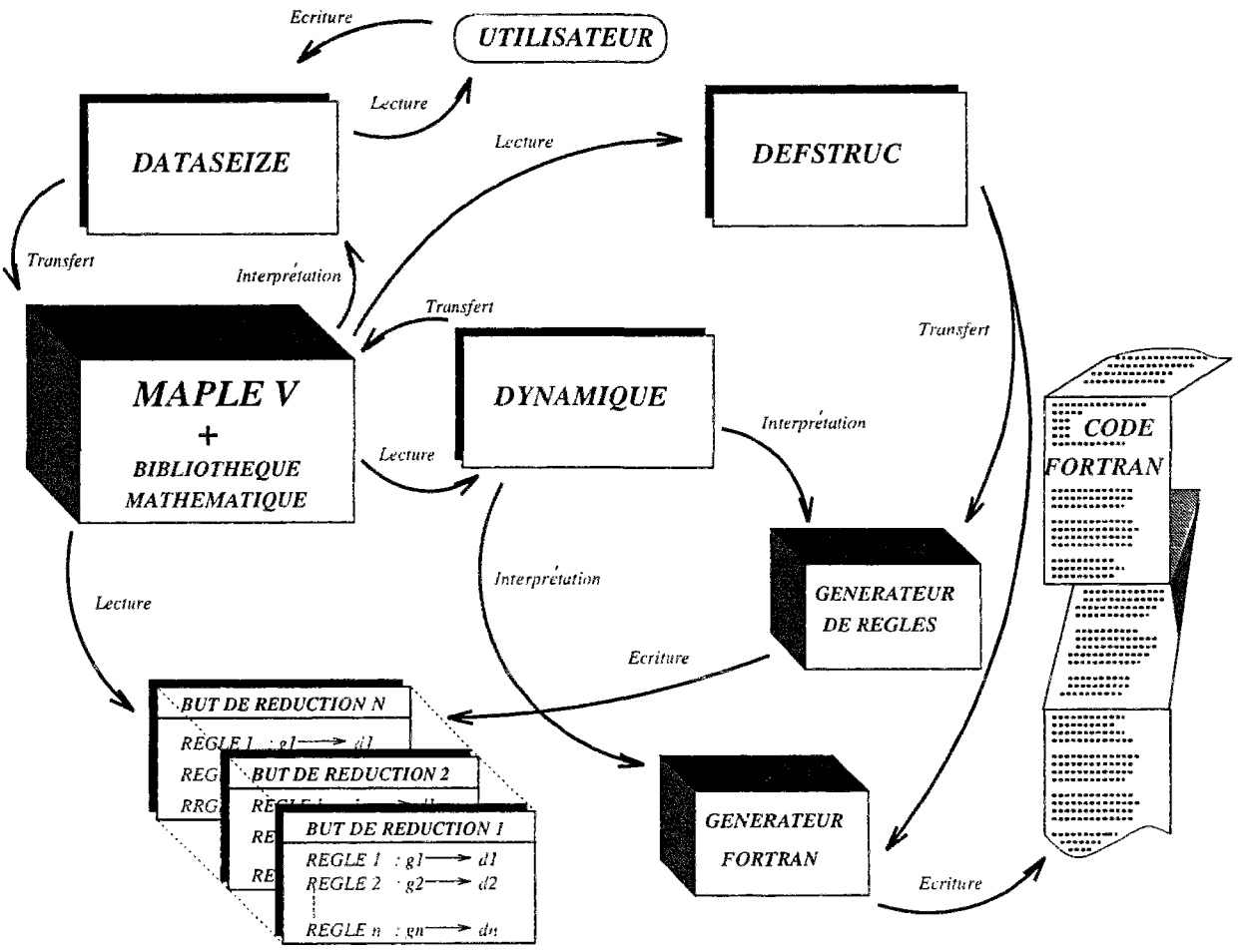
Les principales composantes du système *MEDUSA MF77* sont [Fig. 6.2]:

- le système Maple: qui permet de manipuler les expressions formelles.
- le module *DATASEIZE*: interface Homme–Machine qui permet une saisie correcte des données du problème mécanique.
- le module *DYNAMIQUE*: proposant les schémas de calcul du modèle dynamique selon la partie théorique du chapitre 4 et la simplification automatique, du modèle, relative à la description paramétrée.
- le module *DEFSTRUC* et le générateur de règles qui permettent de structurer le problème dynamique dans des "bases de connaissances".
- un générateur du code Fortran optimisé qui permet, quant à lui, de traduire le code générer à l'étape précédente dans le langage Fortran. Ce code source représente un schéma itératif du calcul optimisé en nombre d'opérations élémentaires et en évaluation des fonctions trigonométriques (les fonctions trigonométriques sont évaluées à des étapes antérieures du code en ligne).

6.1.1 Le système de calcul formel Maple

Il existe actuellement sur le marché de nombreux systèmes de calcul formel[1] (SCF), pour la plupart et depuis plusieurs années disponibles sur un grand nombre d'ordinateurs, dont: Axiom,

[1]Dans la suite nous allons noter les systèmes de calcul formel par les initiales SCF.



Cette figure schématise la nature des liaisons entre les différents composantes du système *MEDUSA MF77*. Ces liens sont les actions de:

- *Lecture*: simple lecture de données ou du module.
- *Ecriture*: écriture de données.
- *Transfert*: transfert de données (transite ⊕ traitement).
- *Interprétation*: interprétation ou réécriture des données symboliques.

Fig. 6.2: Modèle conceptuel de *MEDUSA MF77*

Macsyma, Maple, Mathcade, Mathematica, Reduce, ... etc. Les raisons qui nous ont poussé à

```
->K:=Fraction Polynomial Integer

   (1)  Fraction Polynomial Integer
                                                              Type: Domain
->Dual:=CliffordAlgebra(1,K,quadraticForm 0)
   loading /u/cerma/5/axiom/mnt/rios/algebra/SQMATRIX.NRLIB for domain
   SquareMatrix
   loading /u/cerma/5/axiom/mnt/rios/algebra/MATRIX.NRLIB for domain
   Matrix
   loading /u/cerma/5/axiom/mnt/rios/algebra/VECTOR.NRLIB for domain
   Vector
   loading /u/cerma/5/axiom/mnt/rios/algebra/IIARRAY2.NRLIB for domain
   InnerIndexedTwoDimensionalArray
   loading /u/cerma/5/axiom/mnt/rios/algebra/MATCAT-.NRLIB for domain
   MatrixCategory&
   loading /u/cerma/5/axiom/mnt/rios/algebra/SMATCAT-.NRLIB for domain
   SquareMatrixCategory&
   loading /u/cerma/5/axiom/mnt/rios/algebra/DIRPROD.NRLIB for domain
   DirectProduct
   loading /u/cerma/5/axiom/mnt/rios/algebra/IVECTOR.NRLIB for domain
   IndexedVector
   loading /u/cerma/5/axiom/mnt/rios/algebra/IARRAY1.NRLIB for domain
   IndexedOneDimensionalArray
   loading /u/cerma/5/axiom/mnt/rios/algebra/VECTCAT-.NRLIB for domain
   VectorCategory&
   loading /u/cerma/5/axiom/mnt/rios/algebra/QFORM.NRLIB for domain
   QuadraticForm
   loading /u/cerma/5/axiom/mnt/rios/algebra/RMATCAT-.NRLIB for domain
   RectangularMatrixCategory&

   loading /u/cerma/5/axiom/mnt/rios/algebra/ARR2CAT-.NRLIB for domain
   TwoDimensionalArrayCategory&
   (2)  CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
                                                              Type: Domain
->eps: Dual:=e(1)
   loading /u/cerma/5/axiom/mnt/rios/algebra/CLIF.NRLIB for domain
   CliffordAlgebra
   loading /u/cerma/5/axiom/mnt/rios/algebra/DIRPCAT-.NRLIB for domain
   DirectProductCategory&

   (3)  e
         1
                    Type: CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
->x:=a+eps*b

   (4)  a + b e
               1
                    Type: CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
->y:=c+eps*d

   (5)  c + d e
               1
                    Type: CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
->x+y

   (6)  c + a + (d + b)e
                       1
                    Type: CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
->x*y

   (7)  a c + (a d + b c)e
                         1
                    Type: CliffordAlgebra(1,Fraction Polynomial Integer,MATRIX)
->
```

Dans cette session de travail sur Axiom, on calcul des nombres duaux. Le système Axiom, à la différence des autres systèmes de calcul formel, possède la faculté du calcul typé et, de ce fait, il opère automatiquement au niveau du toplevel toutes les simplifications relatives à ce type de données. L'unité duale ε est noté dans l'output par le symbole e_1.

Fig. 6.3: Session de travail sur Axiom

choisir le système Maple, pour le développement de *MEDUSA MF77*, sont multiples:

- D'abord, Maple est un logiciel de calcul formel et en ce sens il constitue un ensemble d'outils[2] au service du mathématicien.

[2]Il a été développé à l'université de Waterloo à l'Ontario au Canada par "Maple Symbolic Computation Group".

- Il se compose:
  - d'un noyau de très petite taille (inférieur à 400 Ko) comparée aux systèmes Macsyma ou Axiom. Ecrit dans le langage C, il est plus rapide que les logiciels écrits dans des langages interprétés (Macsyma est écrit en Mac-Lisp[3] et Franz-Lisp, alors que Axiom est écrit en Kyoto Common Lisp - KCL).
  - d'une importante librairie mathématique (de 7 à 8 Mo).

- Le système Maple est disponible sur une large variété d'ordinateurs (Micro-ordinateurs, stations de travail, Micro-Vax, Vax ...). Cela va assurer, par la même occasion, la portabilité de toute application basée sur Maple à toutes ces machines.

- Le fait que Maple est livré avec les sources des programmes de sa librairie mathématique écrits dans le langage Maple (95 % du système) permet une grande lisibilité du système. Ainsi, Certaines fonctions du système peuvent être modifiées pour des besoins personnels. En effet, il est parfois très important de modifier certaines fonctionnalités du système afin de mieux les adapter à une application en cours. Cependant cela n'est pas toujours facile et, parfois, on est obligé de modifier tout une série de programmes (cela sous entend que l'on a décortiqué le programme et ressortit les parties de l'algorithme qui peuvent être adaptés autrement) pour modifier une seule fonction. Cela nécessite bien évidement un temps considérable pour la compréhension des programmes et leur modification. Pour illustrer ce propos, l'étude comparative publiée par l'USAF[4] en 1972, peut servir de référence. Selon cette étude, l'écriture d'une nouvelle instruction coûterait 75 $, alors que la modification d'une instruction dans un logiciel coûterait 4000 $, soit à peut près 53 fois plus coûteuse. Cette programmation peut aller, comme c'est le cas dans *MEDUSA MF77*, jusqu'à disparition totale du système de calcul formel et la création d'un nouveau produit, tel que l'utilisateur du produit final n'aurait plus besoin d'aucune connaissance sur l'utilisation du noyau pour la saisie des données des problèmes et l'obtention des résultats.

6.2 Le code Fortran

La génération automatique de codes sources constitue une technique efficace quand il s'agit d'écrire des programme reflétant l'état dynamique de systèmes articulés complexes. En effet, cela permet, d'une part, de laisser cette tâche fastidieuse à la machine et, d'autre part, d'évite le risque de se tromper dans l'écriture de ces programmes.

Nous nous sommes intéressé au code Fortran, non pas par goût à la difficulté ni à la spécificité

[3] Il existe actuellement une version Maxima du système Macsyma écrite complètement en Common Lisp.
[4] "The High Cost of Software", USAF, Monterey, USA, (1972). Cette référence a été citée par B. Meyer et C. Baudoin [MEYE-80],

de ce langage[5] primitif à bien des égards, mais, par souci de répondre à un large besoin dans le monde industriel. Les principales difficultés auxquelles nous devions apporter une solution pour la génération automatique du code itératif en ligne -dans le langage Fortran- peuvent être énumérées dans les règles lexicales suivantes:

i) Une ligne de code est une chaîne de 72 caractères de l'alphabet Fortran.

ii) Les instructions sont écrites en colonnes 7 à 72 sur une ligne initiale et sur au maximum 9 ligne de continuation.

iii) Une ligne initiale contient un espace en colonne 6. les colonnes de 1 à 5 peuvent contenir une étiquette ou alors que des espaces.

iv) Une ligne de continuation contient des espaces en colonnes 1 à 5 et un symbole spécial $ en colonne 6.

v) Une ligne commentaire doit commencer par un symbole spécial c en colonne 1.

vi) Une ligne de commentaire ne peut être suivie d'une ligne de continuation.

vii) La déclaration de l'entête du sous-programme est de la forme:
`subroutine` <nom_identificateur>(<liste des arguments>)

viii) La déclaration de la précision de calcul suit l'instruction de déclaration de l'entête.

ix) Les zones de déclaration des variables communes à d'éventuelles autres sous-programmes complémentaires est de la forme:
`common` / <iden_var > /<liste des variables>

x) Une instruction ne contenant que des espaces n'est pas admise.

xi) La dernière instruction d'un sous programme est l'instruction `end`.

6.3 Difficultés de générer des schémas optimisés par les SCF

Le but étant de générer les équations de la dynamique au terme d'un processus symbolique itératif. Le schéma de ce calcul est décrit essentiellement par l'algorithme 4.7. Ainsi, le processus de calcul symbolique se ramène à un calcul vectoriel et matriciel dans l'espace de dimension trois. Le système Maple permet de faire un tel développement. Cependant, si l'usage classique des systèmes de calcul formel s'avère très efficace pour le développement explicite des équations de la dynamique (par substitution symbolique des termes), la capacité de tels logiciels à générer des *schémas de calcul itératifs optimaux* susceptibles de faire l'objet d'une traduction automatique

[5]En effet ce langage impose des règles de syntaxe très rigides (strictes). Il possède un mécanisme strictement statique. On peut manipuler des variables scalaires et éventuellement des tableaux à un nombre fixe de champs, mais il n'est pas question de déclarer dynamiquement des variables.

dans un langage de programmation usuel -tel que Fortran, C, Ada, ... etc- est très faible[6]. Dans un SCF, une formule mathématique est surtout destinée à faire l'objet de transformations formelles: factorisations diverses, simplifications formelles, dérivation, intégration ... etc. De plus, au bout d'un développement formel, une formule mathématique ne se présente pas nécessairement sous sa représentation d'arbre topologique optimal.

Pour illustrer ce propos, considérons à titre d'exemple le double produit vectoriel de trois vecteurs symboliques. Soient trois vecteurs $a = [a_1\, a_2\, a_3]^t$, $b = [b_1\, b_2\, b_3]^t$ et $c = [c_1\, c_2\, c_3]^t$. Le double produit vectoriel de a, b et c donne:

$$d = a \times (b \times c) = \begin{bmatrix} a_2\,(b_1\,c_2 - b_2\,c_1) - a_3\,(b_3\,c_1 - b_1\,c_3) \\ a_3\,(b_2\,c_3 - b_3\,c_2) - a_1\,(b_1\,c_2 - b_2\,c_1) \\ a_1\,(b_3\,c_1 - b_1\,c_3) - a_2\,(b_2\,c_3 - b_3\,c_2) \end{bmatrix}$$

$$= \begin{bmatrix} a_2\,b_1\,c_2 - a_2\,b_2\,c_1 - a_3\,b_3\,c_1 + a_3\,b_1\,c_3 \\ a_3\,b_2\,c_3 - a_3\,b_3\,c_2 - a_1\,b_1\,c_2 + a_1\,b_2\,c_1 \\ a_1\,b_3\,c_1 - a_1\,b_1\,c_3 - a_2\,b_2\,c_3 + a_2\,b_3\,c_2 \end{bmatrix}$$

La complexité apparente de ce développement est 9 additions et 18 multiplications, alors que la réalisation numérique de ce double produit en deux étapes $d = b \times c$ et $d = a \times \underline{d}$ -où $\underline{d}$ désigne l'ancienne valeur de d- ne nécessite plus que 6 additions et 12 multiplications.

6.3.1 Première critique

Analysons maintenant le comportement des primitives du système Maple pour générer un schéma optimisé. La primitive de Maple qui permet de générer du code Fortran est `fortran` et l'option qui permet d'optimiser un tel code est `optimized`. Considérons, dans un premier temps, la forme parenthésée du vecteur d.

```
    |\^/|     MAPLE V
._|\|   |/|_.  Copyright (c) 1981-1990 by the University of Waterloo.
 \  MAPLE  /   All rights reserved.  MAPLE is a registered trademark of
 <____ ____>   Waterloo Maple Software.
      |        Type ? for help.

> with(linalg):

> a:=array([a1,a2,a3]):

> b:=array([b1,b2,b3]):

> c:=array([c1,c2,c3]):

> d:=crossprod(a,crossprod(b,c));
                d := [ a2 (b1 c2 - b2 c1) - a3 (b3 c1 - b1 c3),
                       a3 (b2 c3 - b3 c2) - a1 (b1 c2 - b2 c1),
                       a1 (b3 c1 - b1 c3) - a2 (b2 c3 - b3 c2) ]

> fortran(d,optimized);
      t4 = b1*c2-b2*c1
```

[6]En fait, ce n'est pas l'ordre d'idée des applications dans lequel ont été conçus ces systèmes.

```
      t9 = b3*c1-b1*c3
      t16 = b2*c3-b3*c2
      d(1) = a2*t4-a3*t9
      d(2) = a3*t16-a1*t4
      d(3) = a1*t9-a2*t16
```

le code est bien optimal puisqu'il présente 6 additions et 12 multiplications.

Considérons maintenant la forme éclatée du vecteur d:

```
> d:=map(simplify,crossprod(a,crossprod(b,c)));
                d := [ a2 b1 c2 - a2 b2 c1 - a3 b3 c1 + a3 b1 c3,
                       a3 b2 c3 - a3 b3 c2 - a1 b1 c2 + a1 b2 c1,
                       a1 b3 c1 - a1 b1 c3 - a2 b2 c3 + a2 b3 c2 ]

>fortran(d,optimized);
      t1 = b1*c2
      t3 = b2*c1
      t6 = b3*c1
      t9 = b1*c3
      t12 = b2*c3
      t14 = b3*c2
      d(1) = a2*t1-a2*t3-a3*t6+a3*t9
      d(2) = a3*t12-a3*t14-a1*t1+a1*t3
      d(3) = a1*t6-a1*t9-a2*t12+a2*t14
```

le code présenté, cette fois, est loin d'être optimal puisqu'il présente 9 additions et 18 multiplications et cela va être le cas de toute expression qui aura subit des transformation avant de faire l'objet d'une traduction dans le code Fortran. Ce comportement bizarre -tout au moins inattendu- de l'option `optimized` peut s'expliquer par le fait qu'une telle opération se base sur le processus de hachage (anglais: hash coding) de l'expression traitée et ne va donner effectivement le code optimal de cette dernière que si l'arbre de l'expression mathématique est sous sa forme optimale (i.e. que si l'expression mathématique est de signature[7] minimale). Les complexités établies dans ces deux cas restent les mêmes même après compilation des fragments de programmes correspondants [Annexe E].

Le lecteur pourra remarquer que dans un cas simple comme celui-ci, un programmeur habitué au langage Fortran écrirait le premier fragment de programme plutôt qu'au deuxième.

6.3.2 Seconde critique

Cette critique liée aux applications précédentes. On remarque qu'il n'y a pas de moyens pour contrôler les noms des variables intermédiaires dans le code généré. Ainsi, l'utilisateur n'a pas la possibilité de les déclarer, par un processus automatique, dans la zone de déclaration des variables au début du code généré.

[7]Nous entendons par signature, le nombre des sous termes quand il s'agit d'un terme somme et le nombre de facteurs lorsqu'il s'agit d'un terme produit.

6.3.3 Troisième critique

La troisième critique est liée à l'évaluation dynamique au toplevel des logiciels de calcul formel. En fait, il n'y a aucun processus naturel à rendre statique une opération. En effet, quand des valeurs sont affectées à des symboles, toute opération sur ces symboles au niveau syntaxique entraîne automatiquement un calcul conséquent sur les valeurs.

6.3.4 Quatrième critique

La primitive `fortran` procède automatiquement (sans qu'une demande expresse de la part de l'utilisateur ne soit faite) à la décomposition d'une expression en instructions intermédiaires. Le fragment de programme résultant ne peut être facilement interprété par un autre utilisateur.

Les solutions proposées dans *MEDUSA MF77* sont de sorte à contourner toutes ces difficultés. Les deux prochaines sections décrivent, sur des exemples simples, un langage formel typé totalement implémenté dans le langage de Maple. Il permet ainsi d'apporter des solutions à ces problèmes et de répondre aux critères qualitatifs des solutions aux problèmes de génération automatique des schémas de calcul optimisés.

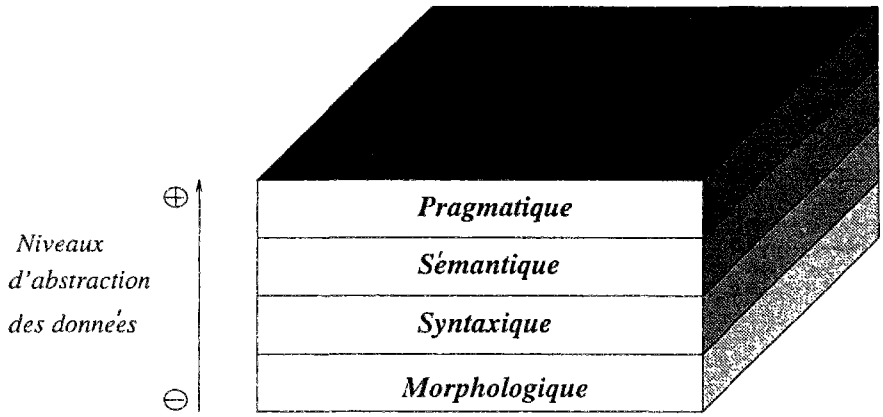
6.4 Création dynamique d'objets dans le système *MEDUSA MF77*

La solution que nous proposons se base sur la programmation orienté objet. En effet, la programmation orientée objet permet de regarder le système comme une collection d'objets qui communiquent entre eux. Ainsi, notre approche sera basée sur le "masquage" des informations manipulées par le système de calcul formel. Ces objets décrivent alors des représentations intermédiaires ou des interprétations selon un certain niveau d'abstraction des données initiales. Au cours du processus de génération automatique des équations de la dynamique, leur ensemble reflète donc l'état de résolution du système *MEDUSA MF77*.

Pour établir le schéma de calcul du modèle dynamique, le système *MEDUSA MF77* utilise les algorithmes établis au chapitre 4, il effectue essentiellement un calcul matriciel et vectoriel. Ainsi, les objets symboliques considérés par *MEDUSA MF77* sont de trois types (ou classes):

- les symboles de type scalaire,
- les symboles de type vecteur,
- les symboles de type matrice carrée.

Comme dans la plupart des logiciels de calcul formel, sous le système Maple, on ne peut utiliser des opérations sur des structures vectorielles ou matricielles -telles que les fonctions: `crossprod`, `add`, `multiply`, `transpose`, ...etc- que si elles sont appliquées à des données de type vecteur ou matrice. Pour pouvoir appliquer dynamiquement de telles fonctions, ce type de représentation oblige à créer



On peut distinguer quatre niveaux d'abstraction des données: le niveau pragmatique, le niveau sémantique, le niveau syntaxique et le niveau morphologique. Chaque niveau d'abstraction rassemble des interprétations des données de même niveau d'abstraction.

Fig. 6.4: Niveaux d'abstraction des données

de nouveaux objets et d'écrire de nouvelles fonctions dans le langage du système Maple. C'est dans cet ordre d'idées que nous avons implémenté le module `defstruc.map`. Totalement écrits dans le langage de Maple, les outils définis dans ce module permettent de créer de façon analogue au système de calcul formel Axiom des données typées [Fig. 6.3]. Cet ensemble de moyens permet à *MEDUSA MF77* de créer dynamiquement des scalaires, des vecteurs ou des matrices à partir des identificateurs qui les symbolisent. Les fonctions liées à la création de ces structures de données sont résumées dans le tableau 6.1.

Tableau 6.1 : *Syntaxe des fonctions*

Fonction	*Syntaxe*	*Domaine de définition*
constructeur de structures	`make_object(iden,class,dim)` l'argument `dim` pour le type scalaire est facultatif il n'a aucun effet.	`iden` = identificateur, `class` $\in$ {`scalar`,`vector`,`matrix`}, `dim` $\in \mathbb{N}^*$
identité des objets	`object(iden)`	`iden` = identificateur
prédicat du type scalaire	`scalarp(iden)`	`iden` = identificateur
prédicat du type vecteur	`vectorp(iden)`	`iden` = identificateur
prédicat du type matrice carrée	`matrixp(iden)`	`iden` = identificateur

6.4.1 Constructeur de classes `make_object`

Un constructeur de *MEDUSA MF77* est un généralement identifié par le préfixe `make`. Il comporte obligatoirement des sélecteurs qui lui permettent d'identifier la classe de l'objet que l'on souhaite créer. Cependant, il n'indique aucune valeur en retour, sinon le nom de la classe de l'objet créé. Le constructeur de classes `make_object` permet d'associer à un symbole une classe de représentation de données.

Exemples

```
> read 'defstruc.map':

> make_object(MM,matrix,5);
                                Class: matrix(5,5)

> object(MM);
              [ MM[1, 1]  MM[1, 2]  MM[1, 3]  MM[1, 4]  MM[1, 5] ]
              [                                                   ]
              [ MM[2, 1]  MM[2, 2]  MM[2, 3]  MM[2, 4]  MM[2, 5] ]
              [                                                   ]
              [ MM[3, 1]  MM[3, 2]  MM[3, 3]  MM[3, 4]  MM[3, 5] ]
              [                                                   ]
              [ MM[4, 1]  MM[4, 2]  MM[4, 3]  MM[4, 4]  MM[4, 5] ]
              [                                                   ]
              [ MM[5, 1]  MM[5, 2]  MM[5, 3]  MM[5, 4]  MM[5, 5] ]
```

la fonction `object` permet donc de visualiser la morphologie d'un tel objet, alors que le prédicat `matrixp` permet au système de tester si un objet est de type matrice:

```
> matrixp(MM);
                                     true
```

Des exemples semblables peuvent être donnés à propos des classes d'objets de type vecteur et scalaire:

```
> make_object(VV,vector,4) ;
                                Class: vector(4)

> object(VV);
                        [ VV[1], VV[2], VV[3], VV[4] ]

> make_object(s,scalar);
                                  Class: scalar

> scalarp(s);
                                      true
```

Jusqu'à nouvel ordre le symbole s sera considéré comme étant de type scalaire.

Ainsi définis, les objets sont structurés par les attributs qu'ils possèdent. Chaque attribut peut alors recevoir une ou plusieurs instanciations (symboliques ou numériques). Cette définition des objets est plus générale que celle des objets Maple. Elle permet de définir dynamiquement des objets. Par contre, les objets Maple sont définis à partir d'instanciations (par l'affectation classique ":= ").

6.4.2 Constructeurs opératoires

Sur ces bases, nous avons implémenté un ensemble d'outils[8] qui permettent d'opérer des objets à partir de leurs identificateurs. Les structures convenables, pour opérer correctement de telles opérations, sont créées de façon dynamique et peuvent dès lors être utilisées dans des calculs ultérieurs comme des objets structurés (i.e. ayant un type bien défini). Les opérations définies sont données dans le tableau 6.2.

Exemples

Le constructeur `make_rotation` permet de créer des rotations suivant les trois axes définis par une famille fondamentale (i.e. un repère). L'angle d'une telle rotation peut être éventuellement sous forme d'une expression formelle:

```
>make_rotation(Rot1,x^2+2*x+3,1);
                                Class: matrix(3,3)

> object(Rot1);
                          [ 1     0         0     ]
                          [                       ]
                          [ 0  cos(%1)  - sin(%1) ]
                          [                       ]
                          [ 0  sin(%1)   cos(%1)  ]

                               2
%1 :=                          x  + 2 x + 3
```

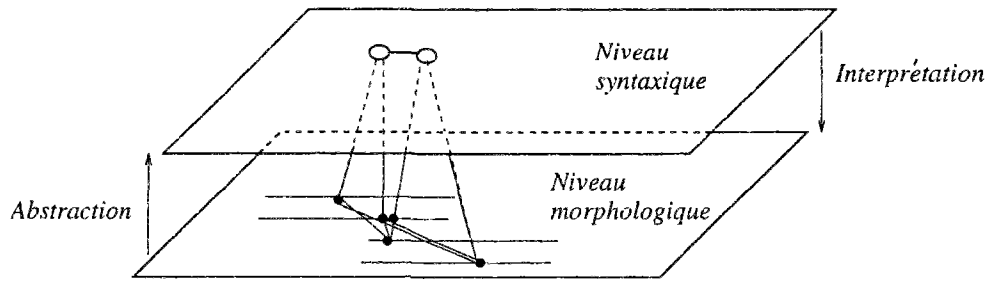
[8] Le terme "constructeurs" dans ce paragraphe ne correspond pas tout à fait à la terminologie classique au sens de la programmation à objets. Les constructeurs ici désignent des constructeurs de symboles au sens du calcul symbolique sur des structures.

Tableau 6.2 : *Syntaxe des constructeurs opératoires*

Fonction	*Syntaxe*	*Domaine de définition*
constructeur de rotations	`make_rotation(iden,ang,axis)`	`iden` = identificateur, `ang` = expression, `axis` $\in \{1,2,3\}$
constructeur de transposées de matrices	`make_transmat(iden,mat,dim)`	`iden,mat` = identificateur, `dim` $\in \mathbb{N}^*$
constructeur de produit de matrices	`make_multmat(iden,mat1,mat2,dim)`	`iden,mat1,mat2` = identificateurs, `dim` $\in \mathbb{N}^*$
constructeur de somme de matrices	`make_addmat(iden,mat1,mat2,dim)`	`iden,mat1,mat2` = identificateurs, `dim` $\in \mathbb{N}^*$
constructeur du produit vectoriel en dimension trois	`make_crosvect(iden,vec1,vec2)`	`iden,vec1,vec2` = identificateurs
constructeur de somme de deux vecteurs	`make_addvect(iden,vec1,vec2,dim)`	`iden,vec1,vec2` = identificateurs, `dim` $\in \mathbb{N}^*$
constructeur du produit d'un vecteur par une matrice	`make_vecparmat(iden,mat,vec,dim)`	`iden,mat,vec` = identificateurs, `dim` $\in \mathbb{N}^*$
constructeur du produit d'une structure par un scalaire	`make_mulscal(iden,iscal,istruc,class,dim)`	`iden,iscal,istruc` = identificateurs, `class` $\in$ {`scalar,vector,matrix`}, `dim` $\in \mathbb{N}^*$

Le constructeur `make_transmat(B,A,4)` crée un objet `B` de type matrice et pointe sa référence vers un autre objet `A` de même classe:

```
> make_transmat(B,A,4);
                              Class: matrix(4,4)

> object(B);
                    [ A[1, 1]  A[2, 1]  A[3, 1]  A[4, 1] ]
                    [                                   ]
                    [ A[1, 2]  A[2, 2]  A[3, 2]  A[4, 2] ]
                    [                                   ]
                    [ A[1, 3]  A[2, 3]  A[3, 3]  A[4, 3] ]
                    [                                   ]
                    [ A[1, 4]  A[2, 4]  A[3, 4]  A[4, 4] ]
```



L'interprétation d'une opération, décrite au niveau syntaxique, se traduit par un nombre d'opérations plus important au niveau morphologique.

Fig. 6.5: Interprétation des opérations

Nous pourrons tenir le même propos au sujet des constructeur `make_addmat`, `make_mulmat`, `make_addvect`, `make_crosvect` et `make_mulscal`. En voici quelques exemples:

```
> make_multmat(T,R,S,2);
                              Class: matrix(2,2)

> object(T);
    [ R[1, 1] S[1, 1] + R[1, 2] S[2, 1]  R[1, 1] S[1, 2] + R[1, 2] S[2, 2] ]
    [                                                                       ]
    [ R[2, 1] S[1, 1] + R[2, 2] S[2, 1]  R[2, 1] S[1, 2] + R[2, 2] S[2, 2] ]

> make_crosvect(V,V1,V2);
                              Class: vector(3)

> object(V);

[V1[2] V2[3] - V1[3] V2[2], V1[3] V2[1] - V1[1] V2[3], V1[1] V2[2] - V1[2] V2[1]]
```

6.5 Abstraction symbolique des objets par "masquage"

Le terme masquage est souvent utilisé en conception des systèmes informatiques et en génie logiciel -comme synonyme de secret- pour indiquer toutes les informations qui concernent l'implémentation d'un module privé (au développeur), sauf quand elles sont déclarées publiques, le module est alors protégé officiellement par un *copyright*. Le sens que nous donnons au masquage ici est tout à fait différent. Il est relatif, en quelque sorte, à la simulation de l'appel par nom au

toplevel du système de calcul formel. *masquer une expression mathématique* reviendra à lui attribuer un nom par réécriture formelle. Ce nom ne sera pas directement interprété par l'interpréteur du toplevel du système de calcul formel. Ainsi, tant que le masquage ne sera pas lever l'expression mathématique sera manipulée symboliquement sous ce nom. Cette technique nous permettra d'élargir le champ d'application des systèmes de calcul formel, à des problèmes de "décision" dans la *génération automatique de programmes*, en introduisant la possibilité de suspendre momentanément des informations au niveau du toplevel du système de calcul formel pour laisser aller à son terme le processus de génération d'un code source quasi-optimal. Le code source décrira dans un langage de programmation (ici ce langage sera Fortran), un schéma de calcul par étapes.

En effet, pour établir des schémas de calcul par étapes, *MEDUSA MF77* a besoin de masquer certains termes, voir même, des objets pour pouvoir s'en servir dans le processus itératif. Cependant, puisqu'il est maladroit de masquer dans un calcul matriciel des termes égaux à 0 ou ± 1 (ces valeurs n'entraînent pas d'opérations supplémentaires), chaque fois que le processus de masquage rencontrera de telles valeurs, il va les laisser intactes. Un ensemble d'outils ont été implémentés pour permettre de telles opérations. La fonction de base qui accorde à *MEDUSA MF77* les transformations décrites ci-avant est essentiellement `rewrite_obj` qui permet de masquer des objets de type scalaire, vecteur ou matrice.

Exemples

Considérons une matrice `A` et un vecteur `V` que nous allons définir dans la suite. Mais décrivons auparavant le processus de masquage et de réécriture des valeurs masquées:

```
> initialize(knowledge):
```

A cette étape, on prépare la base de connaissance qui doit être créée par le générateur de règles à recevoir des informations. Ensuite, définissons la matrice `A`:

```
> A:=array([[0,2,a*x+b],[1,sin(x),c],[0,-1,0]]);
                                [ 0     2    a x + b ]
                                [                    ]
                          A := [ 1  sin(x)     c     ]
                                [                    ]
                                [ 0    -1        0    ]
```

Pour masquer la matrice `A`, il suffit alors de demander sa réécriture:

```
>  rewrite_obj(A);
#  REECRITURE ...     -->   1
                                Class: matrix(3,3)

> make_code_obj(A,double);
                                        Done
```

L'application `make_code_obj` permet alors de générer le code Fortran associé à la définition de `A`:

```
c
c   The parameter A(1,1) is equal to zero
c
      A(1,2) = 2.D0
      A(1,3) = a*x+b
c
c   The parameter A(2,1) is equal to one
c
      A(2,2) = dsin(x)
      A(2,3) = c
c
c   The parameter A(3,1) is equal to zero
c
c   The parameter A(3,2) is equal to minus one
c
c   The parameter A(3,3) is equal to zero
c
```

L'option `double` précise au générateur que le code doit être généré en double précision. Maintenant, l'interpréteur de Maple ne connaît plus la matrice `A` que sous une forme masquée:

```
> object(A);
```

$$\begin{bmatrix} 0 & A[1,\ 2] & A[1,\ 3] \\ 1 & A[2,\ 2] & A[2,\ 3] \\ 0 & -1 & 0 \end{bmatrix}$$

En fait, `rewrite_obj` est un processus de masquage et de génération de code symbolique. Il va masquer l'objet sujet de cette opération et pointer sa référence vers la forme explicite de cet objet dite dans la suite sa référence objective.

Réitérons le même processus pour le vecteur `V`:

```
> V:=array([0,2.55,-1]);
                          V := [ 0, 2.55, -1 ]

> make_code_obj(A,double);
                                 Done

> rewrite_obj(V);
#  REECRITURE ...    -->   2
                          Class: vector(3)
```

Au terme de ce processus, la base de connaissance, généré par *MEDUSA MF77*, relative à ce problème simple contient les règles de réécriture reliant chaque objet masqué à sa référence objective:

```
#
# -------- Knowledge based on systems of rules rewriting termes --------
#
#  The rewrite rule :   R_1
```

```
#
A   :=   array(1 .. 3, 1 .. 3,[(3, 1)=0,(1, 1)=0,(3, 2)=-1,(1, 2)=2,(1, 3)=a*x+
b,(3, 3)=0,(2, 1)=1,(2, 2)=sin(x),(2, 3)=c])   ;
#
#
#  The rewrite rule :   R_2
#
V   :=   array(1 .. 3,[(1)=0,(2)=2.55,(3)=-1])   ;
```

Cette base de connaissance ne sera pas directement exploitée au niveau du toplevel de l'interprète de Maple. Cependant, elle sera contrôlée par un processus de contrôle propre à *MEDUSA MF77*.

Les champs des objets `A` et `V` pointent vers leurs interprétations (i.e. références objectives) grâce des pointeurs notés [voir les règles `R_1` et `R_2` ci-avant]:

- `(i)` avec `i = 1..3` pour l'objet vecteur `V` de dimension 3,
- `(i,j)` avec `i,j = 1..3` pour l'objet matrice `A` de dimension 3×3,

qui ne sont en fait que les champs respectifs des références objectives de la matrice `A` et du vecteur `V`. Les règles de réécriture `R_1` et `R_2` traduisent en réalité des systèmes de règles de réécriture, au niveau morphologique, induits par le masquage des objets `A` et `V`:

Tableau 6.3 : *Systèmes de réécritures correspondants*

Règle de réécriture objet	*Système de réécriture champ*
Règle `R_1`	$S_1 \equiv \begin{cases} \texttt{A[1,2]} \longrightarrow \texttt{2} \\ \texttt{A[1,3]} \longrightarrow \texttt{a*x+b} \\ \texttt{A[2,2]} \longrightarrow \texttt{sin(x)} \\ \texttt{A[2,3]} \longrightarrow \texttt{c} \end{cases}$
Règle `R_2`	$S_2 \equiv$ `V[2]` $\longrightarrow$ `2.55`

Les objets `A` et `V` peuvent faire l'objet de toutes sortes de manipulations symboliques. Etablissons, par exemple, le code associé au produit de `V` par`A`:

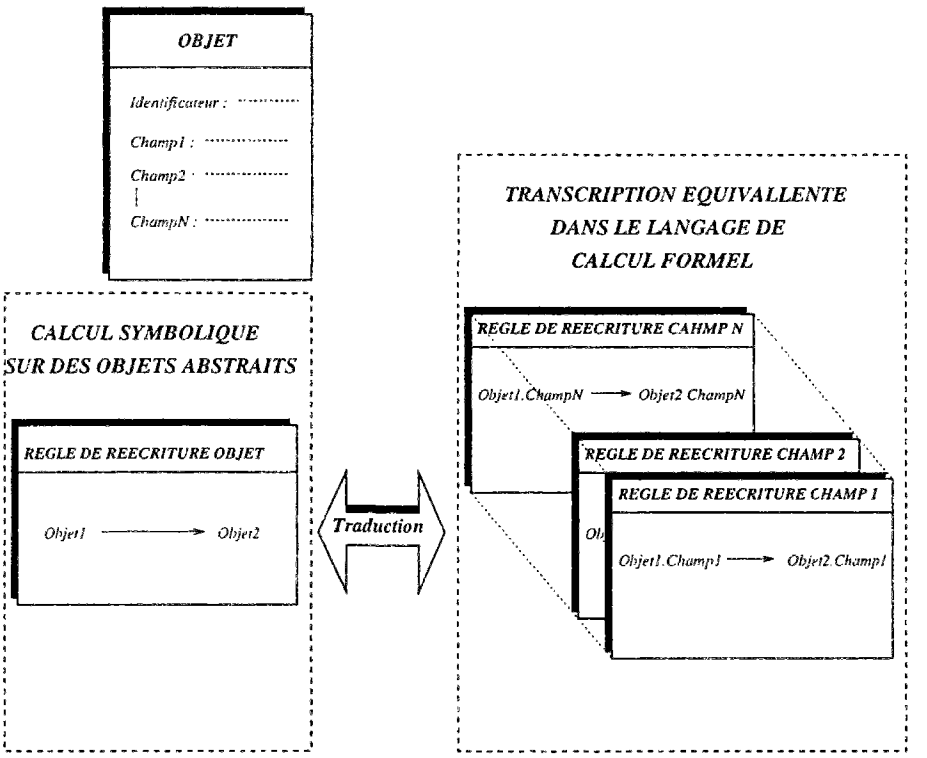
```
> W:= multiply(A,V):

>  make_code_obj(W,double);
                                    Done
```

Une fois, on n'a plus besoin de cette représentation abstraite, le système d'inférence peut être déclenché pour restaurer au toplevel les informations masquées. La fonction qui permet le déclenchement du système d'inférence est `control`:

```
> control(knowledge);
                                    Done
```

la "valeur symbolique" de l'objet `W` devient donc:

```
> object(W);
                    [ 5.10 - a x - b, 2.55 sin(x) - c, -2.55 ]
```



Un objet peut être vu comme une collection de champs. Ainsi, une règle de réécriture sur des objets transcrit un système de règles de réécriture champs. Notre système de génération de règles génère des règles de réécriture sur des objets écrit dans le langage de Maple. La traduction des règles objets en règles champs est établit dans la base de connaissance dans le langage de l'interprète du système Maple.

Fig. 6.6: Traduction de la règle objet en règles champs

qui n'est autre que le produit du vecteur V par la matrice A.

Ce processus a permis de généré le code Fortran en ligne associé au schéma de calcul du produit de V par A:

```
c
c   The parameter A(1,1) is equal to zero
c
      A(1,2) = 2.D0
      A(1,3) = a*x+b
c
```

```
c   The parameter A(2,1) is equal to one
c
      A(2,2) = dsin(x)
      A(2,3) = c
c
c   The parameter A(3,1) is equal to zero
c
c   The parameter A(3,2) is equal to minus one
c
c   The parameter A(3,3) is equal to zero
c
c   The parameter V(1) is equal to zero
c
      V(2) = 0.255D1
c
c   The parameter V(3) is equal to minus one
c
      W(1) = A(1,2)*V(2)-A(1,3)
      W(2) = A(2,2)*V(2)-A(2,3)
      W(3) = -V(2)
```

Les paramètres nuls ou égaux à ± 1 sont placés en commentaires pour permettre à tout autre utilisateur du code généré de connaître leur valeurs exactes. Par contre, elles sont prises en compte dans le développement du schéma de calcul.

Par ailleurs, ce processus va permettre, dans les problèmes de calcul du modèle dynamique du système articulé, de générer automatiquement un code quasi-optimal tout en contrôlant les noms des variables. Il permettra aussi d'avoir des schémas de calcul clairs et commentés. Ainsi, les programmes générés seront plus compréhensibles. Leur modification à la main pour une éventuelle réutilisation sera désormais possible.

6.6 Module de saisie des données dans *MEDUSA MF77*

6.6.1 Principe de la description paramétrée

Pour faciliter[9] l'utilisation de *MEDUSA MF77*, dans des problèmes de génération du code nécessaire au calcul du modèle dynamique, nous avons écrit un interface permettant la saisie des données mécaniques du problème. Le module *DATASEIZE* permet de saisir la description parametrée d'un problème spécifique dans un mode totalement interactif. Le système *MEDUSA MF77* va guider l'utilisateur par un système de "Questions/Réponses" à fournir une description paramétrée du robot manipulateur aussi complète que possible. Toutefois, quand l'utilisateur ignore certaines informations demandées par le système, il a la possibilité de répondre par "auto".

[9]Une étude publié au rapport final de la délégation à la recherche et à l'innovation auprès du ministère de l'équipement à révélée que le profil des utilisateurs des logiciels chez certains organismes clients, ingénieurs et techniciens, vont des plus compétents aux très faibles [Etude de Définition d'une Stratégie de Valorisation de Deux Produit Logiciels, Marché N° 91 01073 00 223 75 01].

Cette commande a pour effet d'indiquer à *MEDUSA MF77* qu'elle doit elle même générer automatiquement une variable correspondant à la donnée en cours de saisie. La spécification d'une donnée peut être de nature:

- Numérique: auquel cas, l'utilisateur deverra spécifier la valeur numérique de cette donnée.
- Symbolique: là encore, il y a possibilité de faire deux choix:
  - l'utilisateur peut imposer l'identificateur symbolique de la donnée en question selon son propre choix.
  - l'utilisateur peut laisser au système le soin de créer dynamiquement la variable correspondante. Il doit répondre par "`auto`".

La description paramétrée proposée par le module *DATASEIZE* consiste alors à spécifier:

- L'identificateur commun des nom de fichiers que doit créer le système *MEDUSA MF77* sur le disque de la machine hôte.
- Le nombre de degrés de liberté du robot manipulateur.
- La nature des articulations de la structure articulée.
- Prise en compte ou non de l'effet de grvité.
- Sens de l'axe normal dirigé vers le centre de la terre par rapport aux axes de la famille fondamentale de référence.
- Donnée des paramètres de Denavit–Hartenberg pour la description de la géométrie de la structure articulée.
- Spécification des masse des corps.
- Caractérisation ou non des centres de masses des différents éléments de la structure articulé.
- Caractérisation des tenseurs d'inertie au niveau du centre de masse.
- Demande de la génération ou non des équations explicites de la dynamique.

6.6.2 Langage de description – Règles syntaxiques

Par définition un langage est un ensemble de caractères, de symboles et de règles qui permettent d'assembler ces caractères. Ainsi, l'ensemble des caractères définit l'alphabet du langage et la concaténation des caractères permet de former des mots. Cela constitue un vocabulaire. Mais, dans un langage les mots sont disposés sous forme d'énoncés:

- Cette disposition est décrite par les règles de syntaxe qui expriment les rapports entre les caractères de l'alphabet.

- Ces énoncés doivent avoir une signification sémantique traduisant leurs rapports avec le réel.

Le système *MEDUSA MF77* pose des question précises à l'utilisateur. La réponse à ces questions ne nécessite aucun préaquis du savoir faire en programmation informatique ni aucune connaissance sur les fonctionnalités et les fonctions de base du logiciel de calcul formel Maple. Cependant pour pouvoir répondre convenablement aux questions de *MEDUSA MF77*, il faut respecter un nombre de règles de syntaxes propres aux système *MEDUSA MF77*. Cette syntaxe peut être exprimeée dans les notations EBNF [10] (Extended Backus–Naur Form). Les règles données par le tableau 6.4 [page 191] expriment une version simplifiée de la syntaxe du langage de la description paramétrée proposée dans le module *DATASEIZE*. Pour construire un énoncé syntaxiquement valide dans le langage de saisie de *MEDUSA MF77*, il faut nécessairement passer par un nombre de ces règles.

6.7 Calcul du modèle dynamique optimisé

Pour optimiser le nombre d'opérations élémentaires -en additions, multiplications et évaluations des fonctions trigonométriques- *MEDUSA MF77* procède à la génération d'un nombre de variables intermédiaires.

6.7.1 variables relatives aux déplacements de passage entre corps adjacents

Elles découlent du calcul symbolique des parties réelles et duales des matrices duales associées aux déplacements entre corps adjacents:

variables relatives à la partie réelle d'une matrice de passage

La partie réelle de la matrice $[\boldsymbol{D}_i^{i-1}{}_*]$ n'est autre que la matrice:

$$D_i^{i-1} = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \cos(\gamma_i)\sin(\theta_i) & \cos(\gamma_i)\cos(\theta_i) & -\sin(\gamma_i) \\ \sin(\gamma_i)\sin(\theta_i) & \sin(\gamma_i)\cos(\theta_i) & \cos(\gamma_i) \end{pmatrix}$$

pour optimiser le nombre d'opérations élémentaires et l'évaluation des fonctions sinus et cosinus à chaque multiplication d'un vecteur par la matrice D_i^{i-1} ou sa transposée dans le processus itératif

[10] Dans de telles notations à chaque règle est associée une variable syntaxique notée $< \cdots >$. Cette variable va dénotée chaque fois qu'elle sera utilisée l'ensemble des énoncés qu'elle décrit. Chaque règle va pouvoire être lue comme une expression. Ainsi:

- $< a >::= < expr >$ se lit "la variable $< a >$ se définit comme étant l'expression $< expr >$".
- $< a >< b >$ se lit "$< a >$ suivi de $< b >$" (produit de concaténation).
- $< a >|< b >$ se lit "$< a >$ ou $< b >$" (alternative de cas).
- $< a > \{< b >\}$ se lit "$< a >$ suivi d'aucune ou plusieurs variables décrites par $< b >$".

Tableau 6.4 : *Syntaxe du langage de la description paramétrée de DATASEIZE*

Règle 1:	< RÉPONSE >::= < UNE RÉPONSE >< FIN >
Règle 2:	< FIN >::= ; | :
Règle 3:	< UNE RÉPONSE >::= auto | o | n | < NOM COMMUN DES FICHIERS >| < NOMBRE DE D.D.L. >| < NATURE DE L'ARTICULATION >| < PARAMÈTRE DE D–H >| < QUANTITÉ INERTIELLE >| < SENS DE LA NORMALE >
Règle 4:	< NOM COMMUN DES FICHIERS >::= < IDENTIFICATEUR >
Règle 5:	< IDENTIFICATEUR >::= < LETTRE > {< LETTRE >} | < LETTRE > {< LETTRE >} < ENTIER SANS SIGNE >
Règle 6:	< ENTIER SANS SIGNE >::= < CHIFFRE > {< CHIFFRE >}
Règle 7:	< CHIFFRE >::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Règle 8:	< LETTRE >::= $a \mid b \mid c \mid \cdots \mid x \mid y \mid z \mid A \mid B \mid C \mid \cdots \mid X \mid Y \mid Z$
Règle 9:	< NOMBRE DE D.D.L. >::= < ENTIER SUPÉRIEUR À 2 > {< CHIFFRE >}
Règle 10:	< ENTIER SUPÉRIEUR À 2 >::= 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Règle 11:	< NATURE DE L'ARTICULATION >::= prismatique | pivot
Règle 12:	< PARAMETRE DE D–H >::= < EXPRESSION >
Règle 13:	< EXPRESSION >::= < TERME SANS SIGNE > {< TERME AVEC SIGNE >} | < TERME AVEC SIGNE > {< TERME AVEC SIGNE >}
Règle 14:	< TERME AVEC SIGNE >::= < SIGNE >< TERME SANS SIGNE >
Règle 15:	< SIGNE >::= + | −
Règle 16:	< TERME SANS SIGNE >::= < OPÉRANDE SIMPLE > {< OPÉRATION >< OPÉRANDE SIMPLE >}
Règle 17:	< OPÉRATION >::= * | / | ** | ^
Règle 18:	< OPÉRANDE SIMPLE >::= < IDENTIFICATEUR >|< NOMBRE SANS SIGNE >
Règle 19:	< NOMBRE SANS SIGNE >::= *Pi* |< ENTIER SANS SIGNE >| < ENTIER SANS SIGNE >< POINT > {< CHIFFRE >}
Règle 20:	< POINT >::= .
Règle 21:	< QUANTITÉ INERTIELLE >::= < EXPRESSION >
Règle 22:	< SENS DE LA NORMALE >::= −3 | −2 | −1 | 1 | 2 | 3

des algorithmes proposés au chapitre 3, on propose les renommages suivants:

$$\mathtt{CTE}_i = \cos(\theta_i) \quad , \quad \mathtt{STE}_i = \sin(\theta_i)$$

$$\mathtt{CGA}_i = \cos(\gamma_i) \quad , \quad \mathtt{SGA}_i = \sin(\gamma_i)$$

$$\mathtt{CCG}_i = \mathtt{CTE}_i * \mathtt{CGA}_i \quad , \quad \mathtt{SCG}_i = \mathtt{STE}_i * \mathtt{CGA}_i$$

$$\mathtt{SSG}_i = \mathtt{STE}_i * \mathtt{SGA}_i \quad , \quad \mathtt{CSG}_i = \mathtt{CTE}_i * \mathtt{SCG}_i$$

ainsi, la matrice D_i^{i-1} s'écrit:

$$D_i^{i-1} = \begin{pmatrix} \mathtt{CTE}_i & -\mathtt{STE}_i & 0 \\ \mathtt{SCG}_i & \mathtt{CCG}_i & -\mathtt{SGA}_i \\ \mathtt{SSG}_i & \mathtt{CSG}_i & \mathtt{CGA}_i \end{pmatrix}$$

variables relatives à la partie duale d'une matrice de passage

Ecrivons maintenant: $\boldsymbol{L}_i^{i-1} = Du\,[\boldsymbol{D}_i^{i-1}{}_*]$,
on peut écrire:

$$\boldsymbol{L}_i^{i-1} = \begin{pmatrix} -d_i \sin(\theta_i) & -d_i \cos(\theta_i) & 0 \\ \cos(\gamma_i)\, d_i \cos(\theta_i) - l_i \sin(\gamma_i) \sin(\theta_i) & -\cos(\gamma_i)\, d_i \sin(\theta_i) - l_i \sin(\gamma_i) \cos(\theta_i) & -l_i \cos(\gamma_i) \\ \sin(\gamma_i)\, d_i \cos(\theta_i) + l_i \cos(\gamma_i) \sin(\theta_i) & -\sin(\gamma_i)\, d_i \sin(\theta_i) + l_i \cos(\gamma_i) \cos(\theta_i) & -l_i \sin(\gamma_i) \end{pmatrix}$$

On peut réécrire $\boldsymbol{L}_i^{i-1}$:

$$\boldsymbol{L}_i^{i-1} = \begin{pmatrix} \mathtt{L11}_i & \mathtt{L12}_i & 0 \\ \mathtt{L21}_i & \mathtt{L22}_i & \mathtt{L23}_i \\ \mathtt{L31}_i & \mathtt{L32}_i & \mathtt{L33}_i \end{pmatrix}$$

avec:

$$\begin{array}{llll} \mathtt{L11}_i = & -\mathtt{STE}_i * \mathtt{d}_i & , & \mathtt{L12}_i = -\mathtt{CTE}_i * \mathtt{d}_i \\ \mathtt{L21}_i = & -\mathtt{l}_i * \mathtt{SSG}_i + \mathtt{CCG}_i * \mathtt{d}_i & , & \mathtt{L22}_i = -\mathtt{l}_i * \mathtt{CSG}_i - \mathtt{SCG}_i * \mathtt{d}_i \\ \mathtt{L23}_i = & -\mathtt{l}_i * \mathtt{CGA}_i & , & \mathtt{L31}_i = \mathtt{l}_i * \mathtt{SCG}_i + \mathtt{CSG}_i * \mathtt{d}_i \\ \mathtt{L32}_i = & \mathtt{l}_i * \mathtt{CCG}_i - \mathtt{SSG}_i * \mathtt{d}_i & , & \mathtt{L33}_i = -\mathtt{l}_i * \mathtt{SGA}_i \end{array}$$

6.7.2 Autres variables intermédiaire

Résumons sous forme d'un tableau [Tableau 6.5] les noms des variables utilisées par *MEDUSA MF77*, pour la génération du code Fortran issu du calcul symbolique itératif du modèle dynamique:

Tableau 6.5 : *Nomination des variables intermédiaire*

Variables dans les programmes	*Signification des variables dans le texte*
$\mathtt{q}_i$	i^e variable articulaire de position: position généralisée
$\mathtt{Dq}_i$	i^e variable articulaire de vitesse: vitesse généralisée
$\mathtt{DDq}_i$	i^e variable articulaire d'accélération: accélération généralisée
$\mathtt{Qg}_i$	i^e variable articulaire d'effort: effort généralisé
$\mathtt{ros}_i$	le vecteur $\overrightarrow{O_i G_i}$: position du centre de gravité
$\mathtt{om}_i$	le vecteur ω_i: vitesse angulaire
$\mathtt{vl}_i$	le vecteur v_i: vitesse linéaire
$\mathtt{Dom}_i$	le vecteur $\dot{\omega}_i$: accélération angulaire
$\mathtt{W}_i$	le vecteur W_i (i.e. le vecteur "$\dot{v}_i - \vec{g}$")
$\mathtt{Fe}_i$	le vecteur F_i: force inertielle
$\mathtt{Ne}_i$	le vecteur N_i: moment inertiel
$\mathtt{fa}_i$	le vecteur f_i: force articulaire
$\mathtt{na}_i$	le vecteur n_i: moment articulaire
$\mathtt{Ixx}_i$,$\mathtt{Iyy}_i$,$\mathtt{Izz}_i$	les moments autour du centre de masse du corps $\mathcal{C}_i$
$\mathtt{Ixz}_i$,$\mathtt{Ixy}_i$,$\mathtt{Iyz}_i$	les produits de masse associées au corps $\mathcal{C}_i$
gravity	le paramètre local de gravité g

Le système *MEDUSA MF77* utilise d'autres variables auxiliaires pour décrire les découpages des expressions:

Variables intermédiaires pour le calcul des Fe_i

$$VOG_i(1) = vl_i(1) + om_i(2)*ros_i(3) - om_i(3)*ros_i(2)$$

$$VOG_i(2) = vl_i(2) + om_i(3)*ros_i(1) - om_i(1)*ros_i(3)$$

$$VOG_i(3) = vl_i(3) + om_i(1)*ros_i(2) - om_i(2)*ros_i(1)$$

Variables intermédiaires pour le calcul des Ne_i

$$IOM_i(1) = Ixx_i*om_i(1) - Ixy_i*om_i(2) - Ixz_i*om_i(3)$$

$$IOM_i(2) = -Ixy_i*om_i(1) + Iyy_i*om_i(2) - Iyz_i*om_i(3)$$

$$IOM_i(3) = -Ixz_i*om_i(1) - Iyz_i*om_i(2) + Izz_i*om_i(3)$$

Variables intermédiaires pour le calcul des Na_i

$$NFR_i(1) = na_{i+1}(1) + L11_{i+1}*fa_{i+1}(1) + L21_{i+1}*fa_{i+1}(2) + L31_{i+1}*fa_{i+1}(3)$$

$$NFR_i(2) = na_{i+1}(2) + L12_{i+1}*fa_{i+1}(1) + L222*fa_{i+1}(2) + L32_{i+1}*fa_{i+1}(3)$$

$$NFR_i(3) = na_{i+1}(3) + L23_{i+1}*fa_{i+1}(2) + L33_{i+1}*fa_{i+1}(3)$$

L'annexe D traite un exemple complet de génération des équations de la dynamique et du code Fortran optimisé correspondant au calcul numérique du modèle dynamique du robot manipulateur Puma 560.

6.8 Conclusions

Nous avons vu dans ce chapitre comment des techniques d'intelligence artificielle, de réécriture et de programmation orientée objet peuvent contribuer à élargir le champ des applications possibles à partir des logiciels de calcul formel. En particulier, la notion de masquage des informations nous a permis de générer automatiquement des schémas de calcul quasi-optimaux tout en permettant de contrôler les variables intermédiaires nécessaires à optimiser le code. Sur le plan pratique, cette méthode présente l'avantage d'offrir la possibilité de générer automatiquement des schémas de calcul structurés et commentés. Cela permet d'accroître la transparence des calcul et la compréhension des modules. Le générateur des règles de réécriture (liens entre les objets et leurs références objectives) permet à l'aide d'un système de contrôle relativement simple à restaurer l'information masquée à tout instant du processus de calcul formel. Contrairement à la primitive `fortran` les outils définis dans *MEDUSA MF77* permettent de garantir la génération d'un code source quasi-optimal commenté comme nous l'avons déjà évoqué. Toutefois, une utilisation judicieuse des outils définis dans *MEDUSA MF77* permet de bénéficier de toutes les facilités liées aux classes d'objets ainsi que leur fiabilité.

Chapitre 7

MODULE DU CALCUL DUAL

7.1 Introduction

Nous avons écrit le module `dualstruc.map` dans le but de disposer d'un ensemble d'outils permettant d'effectuer des calculs sur des données de type duale (nombres duaux, vecteurs duaux, matrices duales, fonctions de variables duale, construction automatique des prolongements canoniques, ...etc). Actuellement, ce module compte un grand nombre de fonctions *auxiliaires* et un nombre plus restreint de fonctions *principales*. Entièrement implémentées dans le langage Maple, elles peuvent être combinées aux fonctionnalités de base du système Maple pour étendre les possibilités du logiciel à la résolution d'un nouveau type de problèmes (utilisant des données duales). Dans la fenêtre (Maple V Release 2) de la figure ci-après, on présente une session de travail sur Maple où l'on expérimente certaines fonctions développées dans le module `dualstruc.map`. Ce chapitre est consacré à présenter les fonctions *principales* définies dans ce module.

7.2 Présentation des constructeurs duaux

Afin de donner un avant goût de la programmation avec les outils définis dans le module `dualstruc.map`, nous allons présenter des exemples simples, mais qui illustrent déjà quelques différences de style avec le calcul classique sur le système Maple. Les fonctions du module - `dualstruc.map` doivent explicitement être chargées avant utilisation.

```
    |\^/|     MAPLE V
._|\|   |/|_.  Copyright (c) 1981-1990 by the University of Waterloo.
 \  MAPLE  /   All rights reserved.  MAPLE is a registered trademark of
 <____ ____>   Waterloo Maple Software.
      |        Type ? for help.

> read `dualstruc.map`:
```

Comme le module `defstruc.map`, le module `dualstruc.map` dispose d'un constructeur d'objets

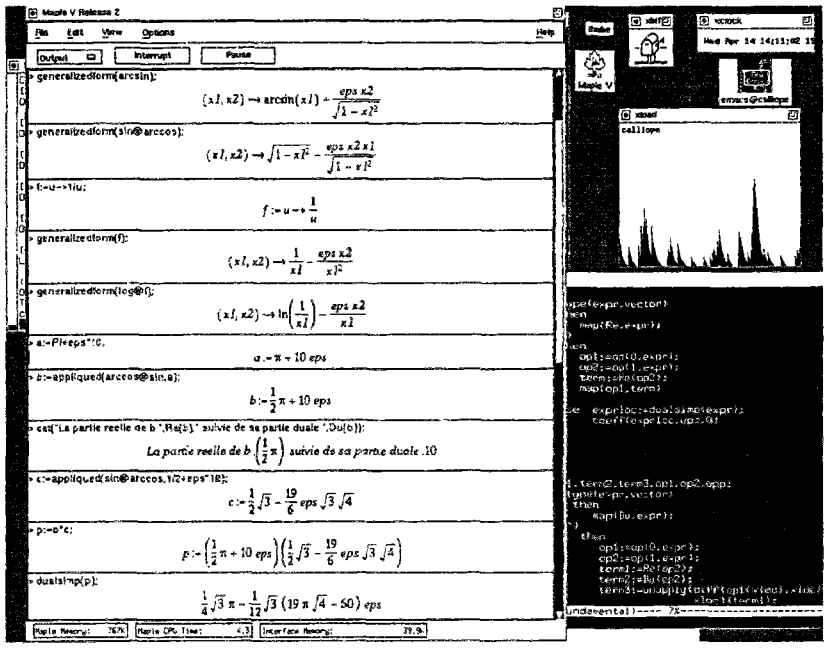


Fig. 7.1: Session de travail sur Maple

matriciels, vectoriels et scalaires. La définition des constructeurs duaux est conforme aux idées développées dans le module `defstruc.map` à la différence près que dans le présent module ces constructeurs permettent de définir de façon autonome des objets de type dual. C'est à dire qu'au niveau morphologique de l'objet, les champs sont des symboles qui représentent des scalaires (i.e. entités élémentaires) de type dual.

Exemple

La manière la plus simple pour construire une matrice duale est de spécifier toutes ses composantes. L'opération principale permettant une telle définition est l'affectation:

```
> Dmat:=array([[1.+7.*eps,e],[3,a+eps*b]]);

                    [ 1. + 7. eps            e          ]
              Dmat := [                                   ]
                    [      3        ra + eps da + eps b ]
```

où le symbole eps désigne l'unité duale ε.

7.2.1 Constructeur de classes

Comme nous l'avons déjà évoqué au chapitre précédent, un constructeur est une fonction d'initialisation que l'on peut appeler (avec des sélecteurs spécifiques) chaque fois que l'on souhaiterait créer une instance d'une classe donnée. Grâce à son constructeur de classes `make_duobj`, le module `dualstruc.map` offre la possibilité de définir des entités duales de façon plus autonome. Le constructeur `make_duobj` permet de créer un objet à partir de son nom et de la spécification de sa nature (classe d'appartenance) et de sa dimension.

La syntaxe d'un tel constructeur est: `make_duobj(iden,struc,dim)`
où:

`iden` : désigne l'identificateur de l'objet qu'on souhaite créer,

`struc` : désigne la structure qu'on souhaite créer,

`dim` : désigne la dimension de l'objet.

La déclaration de la dimension `dim` est inutile lorsque la structure `struc` désigne le type scalaire.

Exemples

```
> make_duobj(A,matrix,3);
                          Class: dual matrix(3,3)

> object(A);
 [ rA[1, 1] + eps dA[1, 1]  rA[1, 2] + eps dA[1, 2]  rA[1, 3] + eps dA[1, 3] ]
 [                                                                           ]
 [ rA[2, 1] + eps dA[2, 1]  rA[2, 2] + eps dA[2, 2]  rA[2, 3] + eps dA[2, 3] ]
 [                                                                           ]
 [ rA[3, 1] + eps dA[3, 1]  rA[3, 2] + eps dA[3, 2]  rA[3, 3] + eps dA[3, 3] ]

> make_duobj(V,vector,2);
                           Class: dual vector(2)

> object(V);
                  [ rV[1] + eps dV[1], rV[2] + eps dV[2] ]

> make_duobj(a,scalar);
                             Class: dual scalar

> object(a);
                                ra + eps da
```

La fonction `object` permet ainsi de visualiser la structure morphologique d'un objet construit à partir de `make_duobj`. Les objets duaux, ainsi créés, comportent une partie réelle dont le nom est un symbole racine préfixée par la lettre `r` et une partie duale préfixée par la lettre `d`.

7.2.2 Opérateur ad d'un torseur

A partir de la donnée des coordonnées duales `X` d'un torseur X, l'opérateur `ad` permet de calculer la matrice duale associée à l'opérateur $ad\,X$ et telle que: $ad\,X\,Y = [X,Y]$ pour tout torseur Y de $\mathfrak{D}$.

Exemples

```
> make_duobj(X,vector,3);
                          Class: dual vector(3)

> EE:=ad(X);
          [          0          - rX[3] - eps dX[3]   rX[2] + eps dX[2]  ]
          [                                                              ]
    EE := [  rX[3] + eps dX[3]           0           - rX[1] - eps dX[1] ]
          [                                                              ]
          [ - rX[2] - eps dX[2]   rX[1] + eps dX[1]           0          ]
```

7.2.3 Déplacements autour des axes d'un repère

La matrice d'un déplacement cylindrique exprimée relativement à une famille fondamentale $\mathfrak{f} = (\xi, \eta, \zeta)$ de l'algèbre de Lie $\mathfrak{D}$ dont l'un des axes est celui du déplacement, s'écrit de façon relativement simple. Le module `dualstruc.map` offre la possibilité de définir de telles matrices. Les fonctions `RD1`, `RD2` et `RD3` permettent de construire les déplacements cylindriques respectivement suivant les premier, second et troisième axes de la famille $\mathfrak{f}$.

Exemples

Si a est le scalaire dual défini dans l'exemple précédent, la *rotation duale*, d'*angle dual* a et d'axe Λ_ξ s'exprime dans la famille fondamentale $\mathfrak{f}$ par:

```
> Dep1:=RD1(a);
              [ 1              0                           0             ]
              [                                                          ]
      Dep1 := [ 0  cos(ra) - eps da sin(ra)  - sin(ra) - eps da cos(ra) ]
              [                                                          ]
              [ 0  sin(ra) + eps da cos(ra)   cos(ra) - eps da sin(ra)  ]

> Dep2:=RD3(Pi/2+eps*c);
                                [ - eps c     -1     0 ]
                                [                      ]
                      Dep2 := [    1     - eps c  0 ]
                                [                      ]
                                [    0        0     1 ]
```

La matrice de tout autre déplacement peut être obtenue à partir des trois matrices `RD1`, `RD2` et `RD3` (grâce à différentes cartes du groupe de Lie $\mathbb{D}$).

7.3 Simplificateurs

Pour mener un processus de calcul formel, étant donné que le but de l'implémentation du module `dualstruc.map` est de pouvoir utiliser, aussi bien des fonctions de base de Maple que des fonctions définies dans ce module, il était nécessaire de concevoir des simplificateurs qui permettent d'appliquer en plus des règles de simplification définies au toplevel, des règles de transformation de type $\varepsilon^2 \longrightarrow 0$. Les simplificateurs les plus importants sont `dualsimp` pour opérer des simplifications sur les nombres duaux, `objdualsimp` pour la simplification des objets duaux, `duwithopt` pour la simplification et l'optimisation des expressions scalaires duales et `objwithopt` pour la simplification et l'optimisation des expressions des champs d'un objet dual quelconque.

Exemples

```
> AEE:=multiply(A,EE);
AEE :=
                                                                    2
[rA[1, 2] rX[3] + rA[1, 2] eps dX[3] + eps dA[1, 2] rX[3] + eps  dA[1, 2] dX[3]

     - rA[1, 3] rX[2] - rA[1, 3] eps dX[2] - eps dA[1, 3] rX[2]

          2
     - eps  dA[1, 3] dX[2], - rA[1, 1] rX[3] - rA[1, 1] eps dX[3]

                                2
     - eps dA[1, 1] rX[3] - eps  dA[1, 1] dX[3] + rA[1, 3] rX[1]

                                                          2
     + rA[1, 3] eps dX[1] + eps dA[1, 3] rX[1] + eps  dA[1, 3] dX[1],

    rA[1, 1] rX[2] + rA[1, 1] eps dX[2] + eps dA[1, 1] rX[2]

          2
     + eps  dA[1, 1] dX[2] - rA[1, 2] rX[1] - rA[1, 2] eps dX[1]

                                2
     - eps dA[1, 2] rX[1] - eps  dA[1, 2] dX[1]]

                                                                    2
[rA[2, 2] rX[3] + rA[2, 2] eps dX[3] + eps dA[2, 2] rX[3] + eps  dA[2, 2] dX[3]

     - rA[2, 3] rX[2] - rA[2, 3] eps dX[2] - eps dA[2, 3] rX[2]

          2
     - eps  dA[2, 3] dX[2], - rA[2, 1] rX[3] - rA[2, 1] eps dX[3]

                                2
     - eps dA[2, 1] rX[3] - eps  dA[2, 1] dX[3] + rA[2, 3] rX[1]

                                                          2
     + rA[2, 3] eps dX[1] + eps dA[2, 3] rX[1] + eps  dA[2, 3] dX[1],

    rA[2, 1] rX[2] + rA[2, 1] eps dX[2] + eps dA[2, 1] rX[2]
```

```
                                                                    2
     + eps  dA[2, 1] dX[2] - rA[2, 2] rX[1] - rA[2, 2] eps dX[1]

                                                  2
     - eps dA[2, 2] rX[1] - eps  dA[2, 2] dX[1]]

                                                                                  2
[rA[3, 2] rX[3] + rA[3, 2] eps dX[3] + eps dA[3, 2] rX[3] + eps  dA[3, 2] dX[3]

     - rA[3, 3] rX[2] - rA[3, 3] eps dX[2] - eps dA[3, 3] rX[2]

          2
     - eps  dA[3, 3] dX[2], - rA[3, 1] rX[3] - rA[3, 1] eps dX[3]

                                  2
     - eps dA[3, 1] rX[3] - eps  dA[3, 1] dX[3] + rA[3, 3] rX[1]

                                                            2
     + rA[3, 3] eps dX[1] + eps dA[3, 3] rX[1] + eps  dA[3, 3] dX[1],

    rA[3, 1] rX[2] + rA[3, 1] eps dX[2] + eps dA[3, 1] rX[2]

          2
     + eps  dA[3, 1] dX[2] - rA[3, 2] rX[1] - rA[3, 2] eps dX[1]

                                  2
     - eps dA[3, 2] rX[1] - eps  dA[3, 2] dX[1]]
```

Dans cet exemple, la matrice duale AEE est définie à partir du produit des matrices duales A et EE. L'opérateur qui a permis une telle opération est la fonction `multiply` (fonction de base de Maple). Visiblement, nous pouvons remarquer qu'un tel opérateur a généré des termes en ε^2. Pour les éliminer, on peut appliquer la fonction `objdualsimp`.

```
> objdualsimp(AEE);
 [(rA[1, 2] dX[3] + dA[1, 2] rX[3] - rA[1, 3] dX[2] - dA[1, 3] rX[2]) eps

      + rA[1, 2] rX[3] - rA[1, 3] rX[2],

     (- rA[1, 1] dX[3] - dA[1, 1] rX[3] + rA[1, 3] dX[1] + dA[1, 3] rX[1]) eps

      - rA[1, 1] rX[3] + rA[1, 3] rX[1],

     (rA[1, 1] dX[2] + dA[1, 1] rX[2] - rA[1, 2] dX[1] - dA[1, 2] rX[1]) eps

      + rA[1, 1] rX[2] - rA[1, 2] rX[1]]

 [(rA[2, 2] dX[3] + dA[2, 2] rX[3] - rA[2, 3] dX[2] - dA[2, 3] rX[2]) eps

      + rA[2, 2] rX[3] - rA[2, 3] rX[2],

     (- rA[2, 1] dX[3] - dA[2, 1] rX[3] + rA[2, 3] dX[1] + dA[2, 3] rX[1]) eps

      - rA[2, 1] rX[3] + rA[2, 3] rX[1],

     (rA[2, 1] dX[2] + dA[2, 1] rX[2] - rA[2, 2] dX[1] - dA[2, 2] rX[1]) eps
```

```
     + rA[2, 1] rX[2] - rA[2, 2] rX[1]]

 [(rA[3, 2] dX[3] + dA[3, 2] rX[3] - rA[3, 3] dX[2] - dA[3, 3] rX[2]) eps

     + rA[3, 2] rX[3] - rA[3, 3] rX[2],

    (- rA[3, 1] dX[3] - dA[3, 1] rX[3] + rA[3, 3] dX[1] + dA[3, 3] rX[1]) eps

     - rA[3, 1] rX[3] + rA[3, 3] rX[1],

    (rA[3, 1] dX[2] + dA[3, 1] rX[2] - rA[3, 2] dX[1] - dA[3, 2] rX[1]) eps

     + rA[3, 1] rX[2] - rA[3, 2] rX[1]]
```

Maintenant, considérons le nombre dual a précédemment défini et le polynôme dual dupoly1 défini par:

```
> dupoly1:=a*(y^2*x^2 + 5*y^2*x + 7*y*x^2) + eps*(4*y*x + x^2 + 2*x):
```

Le simplificateur duwithopt va permettre de simplifier la forme de ce polynôme à deux variables x et y et définir ses parties réelle et duale avec un coût minimum en nombre d'opérations élémentaires:

```
> duwithopt(dupoly1);
    x ra y (7 x + (x + 5) y) + eps x (x + 2 + (4 + 7 da x + (x + 5) da y) y)
```

Le simplificateur objwithopt permet de faire la même opération sur des objet de type scalaire, vecteur ou matrice.

```
> dupoly2:=eps*(x^2+3*x+4):

> W:=array([dupoly1,dupoly2]):

> objwithopt(W);
  [ x ra y (7 x + (x + 5) y) + eps x (x + 2 + (4 + 7 da x + (x + 5) da y) y),
      eps (4 + (3 + x) x) ]
```

de telle fonctions peuvent s'avérer très intéressantes pour la préparation de la génération de codes optimisés (en Fortran par exemple).

7.4 Fonctions

Le module dualstruc.map offre non seulement la possibilité de définir des fonctions d'une (ou plusieurs) variable duale, mais aussi de bénéficier de tous les développements théoriques étudiés dans la partie mathématique de ce mémoire de thèse.

7.4.1 Fonctions parties réelle et duale

Nous avons implémenter les fonctions Re et Du calculant respectivement les parties réelle et duale. Elle peuvent être appliquées aux différents types d'objets.

Exemples

On conserve ici en mémoire les différentes quantités définies dans les exemples précédents:

- La partie duale d'un scalaire dual:

```
> l:=Du(a);
                                    l:= da
```

- La partie réelle de la matrice duale d'une matrice:

```
> Rd:=Re(Dep2);
                                  [ 0  -1  0 ]
                                  [          ]
                             Rd := [ 1   0  0 ]
                                  [          ]
                                  [ 0   0  1 ]
```

Elle représente la matrice de la partie linéaire du déplacement.

- La partie duale du produit de deux matrices duales:

```
>Du(multiply(A,Dep2)) ;
        [ - c rA[1, 1] + dA[1, 2]  - dA[1, 1] - c rA[1, 2]  dA[1, 3] ]
        [                                                            ]
        [ - c rA[2, 1] + dA[2, 2]  - dA[2, 1] - c rA[2, 2]  dA[2, 3] ]
        [                                                            ]
        [ - c rA[3, 1] + dA[3, 2]  - dA[3, 1] - c rA[3, 2]  dA[3, 3] ]
```

7.4.2 Prolongement canonique

Les fonctions essentielles, du module `dualstruc.map`, pour la génération des prolongements canoniques des fonctions de variable réelle sont `generalizedform` qui donne la forme générale d'un prolongement canonique à partir de la fonction de variable réelle qu'on souhaite prolonger et `appliqued` qui applique le prolongement canonique duale d'une fonction de variable réelle à un nombre dual.

Exemples

Les deux fonctions constructrices `generalizedform` et `appliqued` peuvent être appliquées à toute sortes de fonctions de variable réelle. Ainsi dans cette exemple on considère une fonction:

$$g := exp \circ arcsin \circ f \circ sin + log$$

où f est une fonction de variable réelle dont on ignore à priori la forme explicite. Posons:

```
> g:=exp@arcsin@f@sin+log:

> gtild:=generalizedform(g);
     gtild := (x1,x2) -> exp(arcsin(f(sin(x1)))) + ln(x1)

                 /cos(x1) D(f)(sin(x1)) exp(arcsin(f(sin(x1))))     1 \
          + eps x2 |------------------------------------------- + ----|
                 |                              2 1/2               x1 |
                 \             (1 - f(sin(x1)) )                     /
```

Le module `dualstruc.map` offre alors deux possibilités pour appliquer `gtild` à un nombre dual:

- grâce à la fonction `appliqued`:

```
> e:=Pi/4+eps*c :
> appliqued(g,e);
                      1/2
     exp(arcsin(f(1/2 2   ))) + ln(1/4 Pi)

                /     1/2          1/2                    1/2          \
                |    2    D(f)(1/2 2   ) exp(arcsin(f(1/2 2   )))     4 |
          + eps c |1/2 ------------------------------------------ + ----|
                |                              1/2 2 1/2              Pi |
                \                (1 - f(1/2 2   ) )                     /
```

- à partir de la fonction `gtild` elle même:

```
> gtild(Re(e),Du(e));
                      1/2
     exp(arcsin(f(1/2 2   ))) + ln(1/4 Pi)

                /     1/2          1/2                    1/2          \
                |    2    D(f)(1/2 2   ) exp(arcsin(f(1/2 2   )))     4 |
          + eps c |1/2 ------------------------------------------ + ----|
                |                              1/2 2 1/2              Pi |
                \                (1 - f(1/2 2   ) )                     /
```

La solution dans les deux cas est absolument la même.

7.5 Fonctions spéciales

Dans cette rubrique sont classées les constructeurs de ce que nous avons convenu d'appeler dans la partie théorique angles d'Euler duaux et angles de Briant duaux.

7.5.1 Angles d'Euler duaux associés à un déplacement

Les précession, nutation et rotation propres duales pour un déplacement -quelconque, mais, sans singularité- sont données respectivement par les fonctions constructrices `make_phi_eul`, `make_theta_eul` et `make_psi_eul`.

Exemple

En supposant que la matrice duale `A` est la matrice associée à un déplacement donné relativement à une famille fondamentale quelconque:

```
> make_theta_eul(A);
                  2           2 1/2
         (rA[3, 1]  + rA[3, 2] )                    2
  arctan(------------------------) - eps (rA[3, 1]  dA[3, 3]
                 rA[3, 3]

                 2
       + rA[3, 2]  dA[3, 3] - dA[3, 1] rA[3, 1] rA[3, 3]

       - dA[3, 2] rA[3, 2] rA[3, 3])
                                                 /            2           2\
         /           2           2 1/2         2 |    rA[3, 1]  + rA[3, 2] |
        /  ((rA[3, 1]  + rA[3, 2] )    rA[3, 3]  |1 + ---------------------|)
       /                                         |                   2     |
                                                 \          rA[3, 3]       /

> make_psi_eul(A);
              rA[2, 3]    eps (- rA[2, 3] dA[1, 3] + dA[2, 3] rA[1, 3])
        arctan(--------) + ---------------------------------------------
              rA[1, 3]                          2           2
                                       rA[1, 3]  + rA[2, 3]

> make_phi_eul(A);
               rA[3, 2]    eps (- rA[3, 2] dA[3, 1] + dA[3, 2] rA[3, 1])
       - arctan(--------) - ---------------------------------------------
               rA[3, 1]                          2           2
                                        rA[3, 1]  + rA[3, 2]
```

7.5.2 Angles de Briant duaux associés à un déplacement

Les roulis, tangage et lacet duaux pour un déplacement -quelconque, mais, sans singularité- sont données respectivement par les fonctions constructrices `make_phi_bri`, `make_theta_bri` et `make_psi_bri`.

Exemple

```
> make_theta_bri(A);
                  rA[3, 1]
  - arctan(------------------------) - eps (- rA[3, 1] dA[1, 1] rA[1, 1]
                  2           2 1/2
          (rA[1, 1]  + rA[2, 1] )

                                                          2                   2
       - rA[3, 1] dA[2, 1] rA[2, 1] + dA[3, 1] rA[1, 1]  + dA[3, 1] rA[2, 1] )

                                      /                 2      \
         /           2           2 3/2 |          rA[3, 1]       |
        /  ((rA[1, 1]  + rA[2, 1] )    |1 + --------------------|)
       /                               |           2           2|
                                       \    rA[1, 1]  + rA[2, 1] /

> make_psi_bri(A);
              rA[2, 1]    eps (- rA[2, 1] dA[1, 1] + dA[2, 1] rA[1, 1])
        arctan(--------) + ---------------------------------------------
              rA[1, 1]                          2           2
```

```
                                        rA[1, 1]  + rA[2, 1]

> make_phi_bri(A);
                rA[3, 2]    eps (rA[3, 2] dA[3, 3] - dA[3, 2] rA[3, 3])
         arctan(--------) - ------------------------------------------
                rA[3, 3]                          2          2
                                        rA[3, 3]  + rA[3, 2]
```

Il est, bien entendu, clair que l'expression formelle de la matrice `A` peut être aussi complexe qu'on puisse l'imaginer (provenant de la multiplication des matrices de plusieurs déplacements).

Exemple

A titre d'exemple, le tangage dual d'un déplacement obtenu du produit des matrices duales des trois déplacements `RD3(Pi/4+eps)`, `A` et `RD3(Pi/3+eps*c)` est donné par:

```
> DD1:=matdualsimp(multiply(A,RD3(Pi/3+eps*c))):

> DD:=multiply(RD3(Pi/4+eps),DD1):

> make_theta_bri(DD);
                                                 1/2              1/2          2
- arctan(1/8 (2 rA[3, 1] rA[2, 1] rA[2, 2] 3    + rA[3, 2] 3    rA[1, 1]

                                      1/2
    + 2 rA[3, 1] rA[1, 1] rA[1, 2] 3    + 6 rA[3, 2] rA[2, 1] rA[2, 2]

                                                          2
    + 6 rA[3, 2] rA[1, 2] rA[1, 1] + rA[3, 1] rA[2, 1]

                 1/2          2                        2                      2
    + 3 rA[3, 2] 3    rA[1, 2]  + 3 rA[3, 1] rA[1, 2]  + rA[3, 1] rA[1, 1]

                         2              1/2          2
    + 3 rA[3, 1] rA[2, 2]  + rA[3, 2] 3    rA[2, 1]

                 1/2          2    /              2
    + 3 rA[3, 2] 3    rA[2, 2] )  /  (3/4 rA[2, 2]
                                 /

                              1/2                            1/2                2
    + 1/2 rA[2, 1] rA[2, 2] 3    + 1/2 rA[1, 1] rA[1, 2] 3    + 3/4 rA[1, 2]

                 2               2                                        2
    + 1/4 rA[1, 1]  + 1/4 rA[2, 1] )^3/2) - 1/8 eps (4 rA[3, 2] c rA[2, 1]

                 1/2                                    1/2
    - 3 rA[3, 2] 3    dA[1, 2] rA[1, 2] - rA[3, 2] 3    dA[2, 1] rA[2, 1]

               1/2                                      1/2          2
    - rA[3, 2] 3    dA[1, 1] rA[1, 1] - 4 rA[3, 1] c 3    rA[2, 2]

                   1/2          2                                      1/2
    - 4 rA[3, 1] c 3    rA[1, 2]  + 2 dA[3, 1] rA[1, 1] rA[1, 2] 3
```

```
                                      1/2                                    1/2
  + 2 dA[3, 1] rA[2, 1] rA[2, 2] 3    + 4 rA[3, 2] c rA[2, 1] rA[2, 2] 3

                                    1/2             1/2         2
  + 4 rA[3, 2] c rA[1, 1] rA[1, 2] 3    + dA[3, 2] 3    rA[2, 1]

                1/2         2               1/2         2
  + 3 dA[3, 2] 3    rA[2, 2]  + 3 dA[3, 2] 3    rA[1, 2]

  + 6 dA[3, 2] rA[2, 1] rA[2, 2] + 6 dA[3, 2] rA[1, 1] rA[1, 2]

  - 3 rA[3, 2] dA[2, 1] rA[2, 2] - 3 rA[3, 2] dA[2, 2] rA[2, 1]

  - 3 rA[3, 2] dA[1, 1] rA[1, 2] - 3 rA[3, 2] dA[1, 2] rA[1, 1]

                                  1/2                       1/2
  - rA[3, 1] dA[1, 1] rA[1, 2] 3    - rA[3, 1] dA[2, 2] 3    rA[2, 1]

                                  1/2                       1/2
  - rA[3, 1] dA[2, 1] rA[2, 2] 3    - rA[3, 1] dA[1, 2] 3    rA[1, 1]

                1/2                                   1/2         2
  - 3 rA[3, 2] 3    dA[2, 2] rA[2, 2] + dA[3, 2] 3    rA[1, 1]

                     2                       2
  + dA[3, 1] rA[2, 1]  + 3 dA[3, 1] rA[1, 2]  - 3 rA[3, 1] dA[1, 2] rA[1, 2]

  - rA[3, 1] dA[2, 1] rA[2, 1] - 3 rA[3, 1] dA[2, 2] rA[2, 2]

  - 4 rA[3, 1] rA[1, 2] c rA[1, 1] - 4 rA[3, 1] rA[2, 1] c rA[2, 2]

                       2                       2                    2
  + 3 dA[3, 1] rA[2, 2]  + 4 rA[3, 2] c rA[1, 1]  + dA[3, 1] rA[1, 1]

                                  /               2
  - rA[3, 1] dA[1, 1] rA[1, 1]) /  ((3/4 rA[2, 2]
                               /

                            1/2                           1/2                2
  + 1/2 rA[2, 1] rA[2, 2] 3    + 1/2 rA[1, 1] rA[1, 2] 3    + 3/4 rA[1, 2]

                2                2
  + 1/4 rA[1, 1]  + 1/4 rA[2, 1] )^3/2 (1 + 1/64 (

                                  1/2             1/2         2
  2 rA[3, 1] rA[2, 1] rA[2, 2] 3    + rA[3, 2] 3    rA[1, 1]

                                    1/2
  + 2 rA[3, 1] rA[1, 1] rA[1, 2] 3    + 6 rA[3, 2] rA[2, 1] rA[2, 2]

                                                        2
  + 6 rA[3, 2] rA[1, 2] rA[1, 1] + rA[3, 1] rA[2, 1]

                1/2         2                       2                    2
  + 3 rA[3, 2] 3    rA[1, 2]  + 3 rA[3, 1] rA[1, 2]  + rA[3, 1] rA[1, 1]
```

```
                        2              1/2          2
+ 3 rA[3, 1] rA[2, 2]  + rA[3, 2] 3    rA[2, 1]

              1/2          2      /              2
+ 3 rA[3, 2] 3    rA[2, 2] )^2  /  (3/4 rA[2, 2]
                               /

                           1/2                              1/2                2
+ 1/2 rA[2, 1] rA[2, 2] 3    + 1/2 rA[1, 1] rA[1, 2] 3    + 3/4 rA[1, 2]

              2               2
+ 1/4 rA[1, 1]  + 1/4 rA[2, 1] )^3))
```

7.6 Conclusion

Le module `dualstruc.map` définit de puissants outils informatiques de calcul et de manipulation formelle des objets duaux pour le mathématicien. D'autant plus, certains de ces outils sont spécialisés en géométrie et s'appliquent à des problèmes de mécanique. Le but de leur conception est de permettre d'écrire et de manipuler des expressions mathématiques formelles à partir d'objets duaux. L'ensemble de ces outils permet d'exprimer dans une syntaxe simple des modèles de réalités matérielles et physiques très complexes. Ils sont le fruit d'une étude théorique très approfondie des outils mathématiques qu'ils simulent. Ils apportent ainsi au mathématicien, pour ces besoins de modélisation, toutes les performances de rapidité et de précision du calcul sur la machine. Pour l'étudiant et l'utilisateur occasionnel, ils représentent une boîte à outils qui facilite grandement l'utilisation de l'ensemble des moyens de cette représentation mathématique. Le mécanisme d'héritage utilisé permet d'aller encore plus loin dans la programmation sous *MEDUSA MF77* et avec un moindre efforts de programmation.

Partie 4:

ANNEXES

Annexe A

NOTIONS DE LA THEORIE DES GROUPES ET ALGEBRES DE LIE

A.1 Introduction

Un groupe dont les éléments sont étiquetés par un ou plusieurs paramètres continuement variables peut être considéré comme une variété différentielle en prenant pour coordonnées le ou les paramètres d'étiquetage. Cette idée est à l'origine de la théorie des groupes de Lie. Cette théorie, où interviennent des techniques de géométrie, d'algèbre, d'analyse et de topologie, est relativement récente: elle remonte à la fin du siècle dernier.

Le but de cette annexe est d'introduire les définitions et propriétés essentielles de la vaste théorie des groupes algèbres de Lie. Ainsi, elle ne présente aucune originalité. Dans le cadre de ce rappel mathématique, aucune démonstration des résultats énoncés ne sera donnée, sauf s'ils découlent naturellement d'autres relations préétablies dans notre exposé. Toutefois, les lecteurs soucieux des aspects théoriques, pourront se rapporter pour les démonstrations aux références spécialisées abondamment indiquées dans la bibliographie de notre mémoire de thèse.

A.1.1 Variété différentielle – Carte

La notion de variété différentielle consiste à attribuer à un espace topologique de Hausdorff la propriété d'être localement isomorphe à $\boldsymbol{R}^n$.

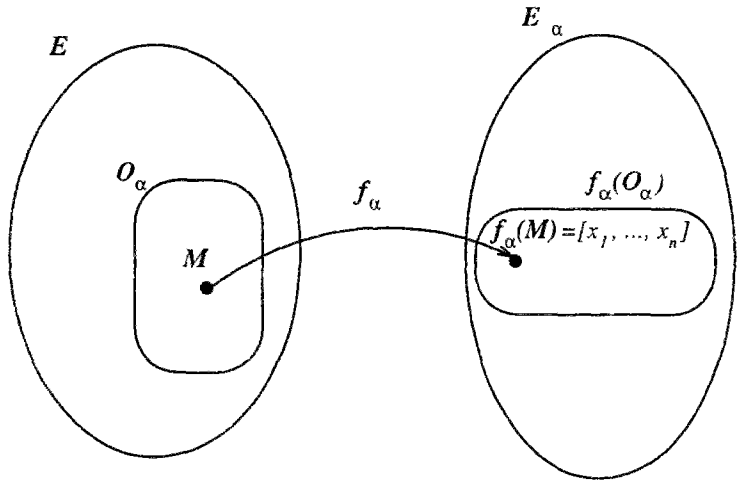
Définition A.1

On appelle carte d'un espace topologique E la donnée d'un ouvert $O_\alpha \in \mathcal{O}_E$ et d'une bijection $f_\alpha : O_\alpha \rightarrow U_\alpha$ (§ Fig. A.1) où U_α est un ouvert d'un espace vectoriel euclidien E_α de dimension finie.

Ainsi, la notion de carte permet de définir des espaces localement euclidiens et par la suite la notion de coordonnées locales.

Définition A.2

On dira que l'espace E est localement euclidien si pour tout point M de E il existe une carte



A un point M de l'ouvert O_α de l'espace E correspond un point $f_\alpha(M)$, du sous ensemble $f_\alpha(O_\alpha)$ de l'espace euclidien E_α, qui est donné par ses coordonnées $[x_1, \ldots, x_n]$ dans E_α. Ainsi on définit une paramétrisation locale de l'espace E sur l'espace euclidien E_α. C'est pour cette raison qu'on dit que E est localement isomorphe à R^n.

Fig. A.1: Carte différentielle

(O_α, f_α) *telle que:*

$$M \in O_\alpha$$

A.1.2 Changement de cartes

Définition A.3

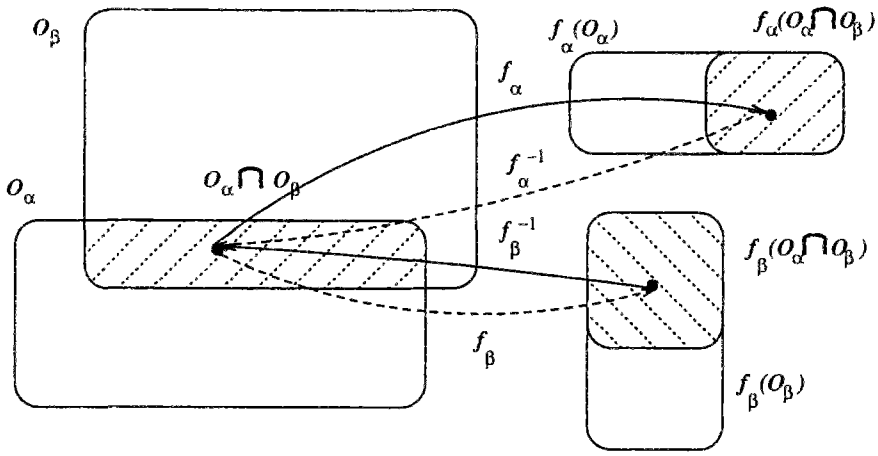
Si (O_α, f_α) et (O_β, f_β) sont deux cartes d'un espace topologique E, on appelle changement de cartes l'application $f_{\alpha,\beta}$ (§ Fig. A.2) définie par la relation:

$$f_{\alpha,\beta} = f_\alpha \circ f_\beta^{-1} \qquad \text{(Eq.A.1)}$$

son domaine est donné par:

$$U_{\alpha,\beta} = f_\beta(O_\alpha \cap O_\beta) \qquad \text{(Eq.A.2)}$$

Remarquons au passage que, si on pose $U_{\beta,\alpha} = f_\alpha(O_\alpha \cap O_\beta)$, alors l'application $f_{\alpha,\beta} : O_{\alpha,\beta} \rightarrow O_{\beta,\alpha}$ définie une bijection entre les ouverts $U_{\alpha,\beta}$ et $U_{\beta,\alpha}$.



En trait continu on peut voir, sur cette figure l'application $f_{\alpha,\beta} : U_{\alpha,\beta} \to U_{\beta,\alpha}$, elle est donnée par $f_\alpha \circ f_\beta^{-1}$ et en trait discontinu son application inverse $f_{\beta,\alpha} : O_{\beta,\alpha} \to O_{\alpha,\beta}$ qui est donnée par $f_\beta \circ f_\alpha^{-1}$.

Fig. A.2: Changement de cartes

A.2 Groupe de Lie

La structure de groupe de Lie confère à un ensemble, à la fois, une structure algébrique de groupe et une structure de variété différentielle.

Définition A.4

On appellera groupe de Lie, un groupe G qui est muni d'une structure de variété différentielle, telle que les opérations du groupe soient différentiables. Ces opérations sont:

(a) d'une part l'opération de composition des éléments du groupe, notée multiplicativement:

$$\begin{array}{rccc} composition: & G \times G & \to & G \\ & (a,b) & \mapsto & a\,b \end{array} \qquad \text{(Eq.A.3)}$$

(b) d'autre part l'opération d'inversion des éléments du groupe:

$$\begin{array}{rccc} inversion: & G & \to & G \\ & a & \mapsto & a^{-1} \end{array} \qquad \text{(Eq.A.4)}$$

A.2.1 Sous-groupes immergés d'un groupe de Lie

Considérons deux groupes de Lie G et G_0 tels que la dimension de G_0 -en tant que variété différentielle- soit au plus égale à celle de G.

Définition A.5

On dira que G_0 est sous-groupe de Lie de G, s'il existe une immersion de G_0 dans G.

A.3 Groupe opérant dans un ensemble

Les groupes de Lie apparaissent le plus souvent comme étant des groupes de transformations d'autres variétés différentielles.

A.4 Action d'un groupe sur un ensemble

Définition A.6

Une action d'un groupe G à gauche sur un ensemble donné $\mathcal{M}$ est un homomorphisme $\phi : G \mapsto Bij(\mathcal{M})$ de G dans le groupe des bijections dans $\mathcal{M}$ tel que[1] *si on note ϕ_g la transformation de $\mathcal{M}$ déterminée par $g \in G$ et e dénote l'élément neutre de G, alors:*

(a) $\phi_e = id_{\mathcal{M}}$,

(b) ϕ_g *est une application bijective:* $\forall g \in G$,

(c) $\phi_g \circ \phi_h = \phi_{g\,h} \quad \forall g, h \in G$

Ainsi, des relations précédentes (a) et (c) on déduit que:

$$\phi_{g^{-1}} = \phi_g^{-1} \tag{Eq.A.5}$$

A.4.1 Fidélité

L'action $(G, \mathcal{M}, \phi)$ est dite fidèle si l'homomorphisme ϕ est injectif.

A.4.2 Transitivité

L'action $(G, \mathcal{M}, \phi)$ est dite transitive si:

$$\forall x, y \in \mathcal{M},\ \exists g \in G :\ \phi_g\, x = y \tag{Eq.A.6}$$

Dans la pratique, il suffit de montrer qu'il existe un élément a de l'ensemble $\mathcal{M}$ tel que:

$$\forall x \in \mathcal{M},\ \exists g \in G :\ \phi_g\, a = x \tag{Eq.A.7}$$

puisque, si $\phi_g\, a = x$ et $\phi_h\, a = y$, alors: $\phi_{h\,g^{-1}}\, x = y$.

L'action $(G, \mathcal{M}, \phi)$ sera dite simplement-transitive si:

$$\forall x, y \in \mathcal{M},\ \exists!\, g \in G :\ \phi_g\, x = y \tag{Eq.A.8}$$

[1] De manière analogue on définit une action à droite d'un groupe sur un ensemble, mais en remplaçant la proposition (c) par:

$$\phi_g \circ \phi_h = \phi_{h\,g} \forall g, h \in G$$

A.4.3 Homogénéité

L'ensemble $\mathcal{M}$ est dit espace homogène s'il existe une opération transitive de G sur $\mathcal{M}$. Il est dit espace homogène sans point fixe s'il existe une opération simplement-transitive de G sur $\mathcal{M}$. On montre que tous les espaces homogènes sans point fixe d'un groupe G sont isomorphes.

A.4.4 Actions d'un groupe de Lie sur lui même

Considérons un groupe Lie G, il agit sur lui même par sa loi de composition interne.

Définition A.7

(a) On définit les translations à gauche de G par:

$$\begin{array}{rccc} L_g: & G & \to & G \\ & a & \mapsto & g\,a \end{array} \qquad \text{(Eq.A.9)}$$

(b) On définit les translations à droite du groupe G par:

$$\begin{array}{rccc} R_g: & G & \to & G \\ & a & \mapsto & a\,g \end{array} \qquad \text{(Eq.A.10)}$$

(c) On définit les automorphismes intérieurs dans G par:

$$\begin{array}{rccc} In_g: & G & \to & G \\ & a & \mapsto & g\,a\,g^{-1} \end{array} \qquad \text{(Eq.A.11)}$$

Les quatre propriétés suivantes découlent immédiatement de la définition précédente:

Propriétés A.1

(a) La composition des translations à gauche de G opère par:

$$L_g \circ L_h = L_{g\,h};\quad \forall g, h \in G \qquad \text{(Eq.A.12)}$$

(b) La composition des translations à droite de G opère par:

$$R_g \circ R_h = R_{h\,g};\quad \forall g, h \in G \qquad \text{(Eq.A.13)}$$

(c) La composition des automorphismes intérieurs de G opère par:

$$In_g \circ In_h = In_{g\,h};\quad \forall g, h \in G \qquad \text{(Eq.A.14)}$$

(d) Les trois actions L_g, R_g et In_g sur G sont liées par la relation suivante:

$$In_g = L_g \circ R_{g^{-1}} \qquad \text{(Eq.A.15)}$$

Ainsi, d'après les relations (Eq.A.12) et (Eq.A.14) les actions L_g et In_g définissent des homomorphismes dans G et d'après la relation (Eq.A.13) l'action R_g définie un anti-homomorphisme dans G.

A.5 Algèbre de Lie d'un groupe de Lie

L'idée générale de l'algèbre de Lie $\mathfrak{g}$ associée à un groupe de Lie G est de construire un champ de vecteurs sur le groupe, à partir d'un vecteur assigné en un point, par des translations à gauche sur le groupe G, lui même.

Soient G un groupe de Lie, g un élément de G et L_g la translation à gauche du groupe G associée à g. Chaque application L_g est un difféomorphisme de la variété G dans elle même. Soit alors L_g^T l'homomorphisme tangent induit par L_g sur l'espace tangent T_gG à la variété G au point g. Comme L est une action de G sur lui même alors:

$$\begin{aligned} L_{g\,h}^T &= (L_g \circ L_h)^T \\ &= L_g^T \circ L_h^T \end{aligned} \qquad \text{(Eq.A.16)}$$

Si e dénote l'élément neutre du groupe de Lie G, on définit alors le vecteur tangent X_g en g comme étant la translation à gauche du vecteur X_e; soit sous forme explicite:

$$X_g = L_g^T X_e; \quad \forall g \in G \qquad \text{(Eq.A.17)}$$

et d'après les relations (Eq.A.17) et (Eq.A.18) on a:

$$\begin{aligned} L_g^T X_h &= \left(L_g^T \circ L_h^T\right) X_e \\ &= L_{g\,h}^T X_e \\ &= X_{g\,h} \end{aligned} \qquad \text{(Eq.A.18)}$$

c'est à dire que l'on translate à gauche X_e de e jusqu'à h, puis de h jusqu'à gh, ou que l'on translate à gauche tout simplement X_e de e jusqu'à gh, le résultat est le même (§ Fig. A.3). En d'autres termes, cela veut dire que le champ de vecteurs X défini par cette construction est indépendant du choix de l'élément par rapport auquel il est défini et que pour toute translation à gauche, on peut écrire:

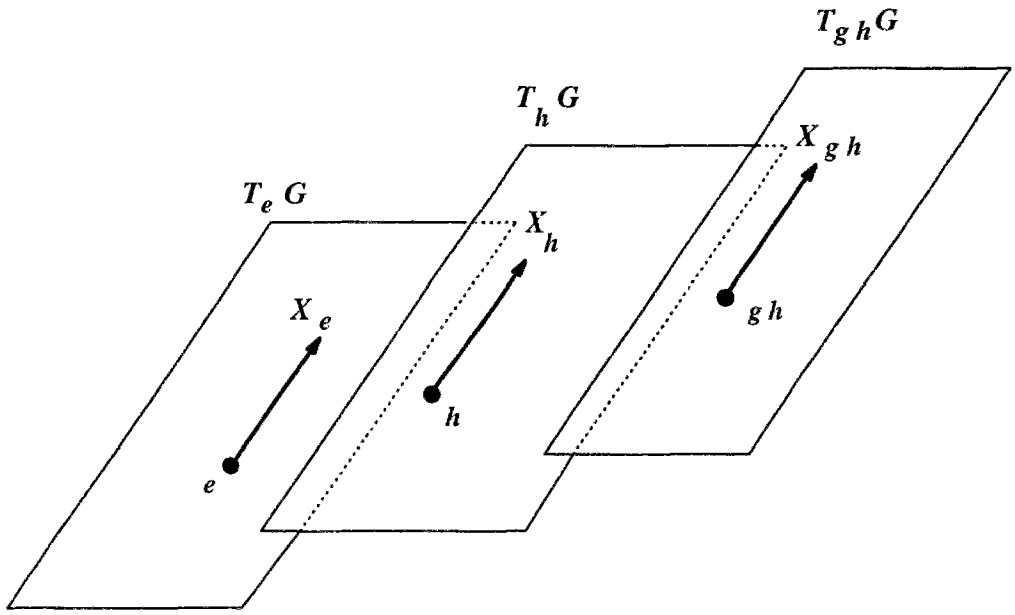
$$L_g^T X = X; \quad \forall g \in G \qquad \text{(Eq.A.19)}$$

Ainsi, le champ de vecteurs X reste invariant par toute translation à gauche.

Inversement soit un champ de vecteurs invariant par translation à gauche, si X_e est sa valeur en e, alors, en vertu de l'invariance à gauche on peut écrire:

$$L_g^T X_e = X_g \qquad \text{(Eq.A.20)}$$

Donc, un champ de vecteurs X invariant par translation à gauche est complètement défini à partir de sa valeur en e.



On translate X_e de e jusqu'à h par L_h^T, puis de h jusqu'à gh par L_g^T.

Fig. A.3: Un champ de vecteurs dans le fibré tangent du groupe G

Soient maintenant deux champs de vecteurs X et Y invariants à gauche et α et β deux nombres réels, alors d'après la linéarité de L_g on peut écrire:

$$\begin{aligned} L_g^T(\alpha X + \beta Y) = &\ \alpha L_g^T X + \beta L_g^T Y \\ = &\ \alpha X + \beta Y \end{aligned} \qquad \text{(Eq.A.21)}$$

c'est à dire que $\alpha X + \beta Y$ est également invariant à gauche. Ainsi, l'invariance à gauche des champs de vecteurs induit une structure d'espace vectoriel, sur $\boldsymbol{R}$, qu'on notera $\mathfrak{g}$. pour l'instant, on appellera cet ensemble espace vectoriel des champs de vecteurs invariants à gauche.

A.5.1 Crochet de Lie

Soit $\mathfrak{g}$ le R-espace vectoriel des champs de vecteurs invariants à gauche sur le groupe de Lie G. Un crochet de Lie $[.,.]$ sur $\mathfrak{g}$ est une application bilinéaire vérifiant les relations:

$$[X,X] = 0, \quad \forall X \in \mathfrak{g} \tag{Eq.A.22}$$

et

$$[X,[Y,Z]] + [Y,[Z,X]] + [Z,[X,Y]] = 0, \quad \forall X,Y,Z \in \mathfrak{g} \tag{Eq.A.23}$$

la relation (Eq.A.23) s'appelle identité de Jacobi. Une conséquence immédiate de la relation (Eq.A.22) est que le crochet est antisymétrique c'est à dire que:

$$[X,Y] = -[Y,X]; \quad \forall X,Y \in \mathfrak{g} \tag{Eq.A.24}$$

Compte tenu de la construction des champs invariants à gauche que nous venons de donner au paragraphe précédent, pour définir le crochet de Lie d'une façon unique, il suffit de le construire, par exemple, sur l'espace tangent T_eG à la variété G en son élément neutre e (pour la structure de groupe de G). Le crochet de Lie sur $\mathfrak{g}$ est induit par la relation:

$$[X,Y]_e = [X_e,Y_e] \tag{Eq.A.25}$$

où X et Y sont des champs de vecteurs déterminés par leurs valeurs X_e et Y_e au point e et on peut écrire:

$$\begin{aligned} L_g^T[X,Y] &= [L_g^T X, L_g^T Y] \\ &= [X,Y] \end{aligned} \tag{Eq.A.26}$$

On appellera donc algèbre de Lie, l'espace vectoriel $\mathfrak{g}$ muni d'un crochet de Lie. Soit $(X^i)_i$ une base de cet espace vectoriel, le crochet de deux éléments de la base est par définition du crochet un élément de $\mathfrak{g}$ et donc s'écrit sous la forme d'une combinaison linéaire des éléments de la base; c'est à dire:

$$[X^i,Y^j] = \sum_k \alpha_k^{i,j} X^k \tag{Eq.A.27}$$

Les nombres $\alpha_k^{i,j}$ ainsi définis s'appellent les constantes de structure de l'algèbre $\mathfrak{g}$ dans la base $(X^i)_i$. La relation (Eq.A.25) se traduit alors par:

$$\alpha_k^{i,j} = -\alpha_k^{j,i} \tag{Eq.A.28}$$

et l'identité de Jacobi entraîne:

$$\alpha_k^{i,j}\alpha_i^{l,m} + \alpha_k^{i,l}\alpha_i^{m,j} + \alpha_k^{i,m}\alpha_i^{j,l} = 0 \tag{Eq.A.29}$$

A.5.2 Sous-groupe à un paramètre

On appelle sous-groupe à un paramètre φ du groupe de Lie G, un difféomorphisme de groupes défini du groupe additif $\boldsymbol{R}$ dans le groupe de Lie G tel que:

$$\varphi(s+t) = \varphi(s)\,\varphi(t) \quad \text{(Eq.A.30)}$$

$$\varphi(0) = e \quad \text{(Eq.A.31)}$$

Soient $\mathcal{M}$ une variété différentielle, G un groupe de Lie opérant à gauche de $\mathcal{M}$ et φ un groupe à un paramètre de G, alors φ définit un groupe à un paramètre de difféomorphismes de $\mathcal{M}$ caractérisé par la relation:

$$\varphi_t(x) = \varphi(t)\,x \quad \text{(Eq.A.32)}$$

En particulier lorsque $\mathcal{M}$ est égale à G lui même, alors les actions $R_{\varphi(t)}$, $L_{\varphi(t)}$ et $In_{\varphi(t)}$ définissent des sous-groupes à un paramètre de $\mathcal{D}iff(G)$ (groupe des difféomorphismes de G).

A.5.3 Application exponentielle

Soient un groupe de Lie G, son algèbre de Lie $\mathfrak{g}$ et un groupe à paramètre φ. On peut construire donc une application exponentielle en posant:

$$exp(X) = \varphi_X(1); \quad \forall X \in \mathfrak{g} \quad \text{(Eq.A.33)}$$

où φ_X est un sous-groupe à un paramètre de G. Enfin, par changement du paramétrage nous obtenons:

$$exp(t\,X) = \varphi_X(t);\ \forall t \in \boldsymbol{R}, \forall X \in \mathfrak{g} \quad \text{(Eq.A.34)}$$

A partir des propriétés des sous-groupes à un paramètre, on déduit les trois relations suivantes:

$$exp(s\,X)\,exp(t\,X) = exp((s+t)\,X) \quad \text{(Eq.A.35)}$$

$$[exp(X)]^{-1} = exp(-X) \quad \text{(Eq.A.36)}$$

$$exp(0) = e \quad \text{(Eq.A.37)}$$

qui caractérisent toutes applications exponentielles ainsi construites.

L'application exponentielle exp de $\mathfrak{g}$ dans G définit un difféomorphisme analytique au voisinage de $0_{\mathfrak{g}}$ (zéro de $\mathfrak{g}$) sur un voisinage de l'élément neutre e du groupe de Lie G.

L'espace tangent $T_{0_{\mathfrak{g}}}\mathfrak{g}$ à l'espace vectoriel $\mathfrak{g}$ en son zéro $0_{\mathfrak{g}}$ peut être identifié à $\mathfrak{g}$. De même l'espace tangent T_eG au groupe de Lie G en son élément neutre e peut être identifié à l'algèbre de Lie $\mathfrak{g}$. Donc, l'application exp^T induite par l'application exponentielle en $0_{\mathfrak{g}}$ définit une application de $\mathfrak{g}$ dans lui même.

A.5.4 Homomorphismes de groupes et d'algèbres de Lie

L'idée de l'application exponentielle peut être introduite afin de simplifier les résultats concernant les homomorphismes. D'abord si $\psi : G \rightarrow H$ est un homomorphisme de groupes de Lie et $exp(t\,X)$ un sous-groupe à un paramètre de G, alors, $\psi\,(exp(t\,X))$ est un groupe à un paramètre de H. De plus:

$$\begin{aligned} \psi(R_{exp\,(t\,X)}\,g) &= \psi(g\,exp\,t\,X) \\ &= \psi(g)\,\psi(exp\,t\,X) \\ &= R_{\psi(exp\ t\,X)}\,\psi(g) \end{aligned} \qquad \text{(Eq.A.38)}$$

d'ou la relation:

$$\psi \circ R_{exp\ t\,X} = R_{\psi(exp\ t\,X)} \circ \psi \qquad \text{(Eq.A.39)}$$

En conséquence, le générateur des translations à droite $R_{\psi(exp\ t\,X)}$ est ψ-lié à X. On notera $\psi^T X$ ce générateur, alors ψ^T est une application des champs de vecteurs invariants à gauche du groupe de Lie G vers les champs invariants à gauche du groupe de Lie H. L'application ψ^T définit donc une application linéaire de l'algèbre de Lie $\mathfrak{g}$ dans l'algèbre de Lie $\mathfrak{h}$ telle que:

$$\psi^T\,[X,Y] = [\psi^T\,X, \psi^T\,Y] \qquad \text{(Eq.A.40)}$$

Ainsi, $\psi^T : \mathfrak{g} \rightarrow \mathfrak{h}$ est un homomorphisme d'algèbre de Lie. De plus elle vérifie la relation:

$$\psi(exp\,t\,X) = exp(t\,\psi^T\,X);\ \forall X \in \mathfrak{g} \qquad \text{(Eq.A.41)}$$

donc finalement:

$$\psi \circ exp = exp \circ \psi^T \qquad \text{(Eq.A.42)}$$

Cette relation peut alors se mettre sous la forme du diagramme suivant:

$$\begin{array}{ccc} \mathfrak{g} & \xrightarrow{\psi^T} & \mathfrak{h} \\ exp\downarrow & & \downarrow exp \\ G & \xrightarrow{\psi} & H \end{array}$$

A.5.5 Coordonnées canoniques

La fonction exponentielle permet de construire des cartes sur le groupe de Lie G en tant que variété différentielle, telles que, si on note $\vartheta(e,G)$ l'ensemble des voisinages de l'élément neutre e dans G, $\vartheta(a,R^n)$ l'ensemble des voisinages de $a \in R^n$ et si $(X_i)_{i=1..n}$ désigne une base de l'algèbre de Lie $\mathfrak{g}$, alors:

- **Coordonnées de première espèce**: Plusieurs définition peuvent être données. Pour notre part, nous considérerons que ce sont les cartes de la forme:

$$q \mapsto exp(q_1 X_1 + \cdots + q_n X_n) \circ A$$

où A est un élément fixe du groupe G et q décrit un voisinage de 0 dans $\boldsymbol{R}^n$. On obtient ainsi une carte au voisinage de l'élément A.

- **Coordonnées de seconde espèce**: Ce sont les cartes de la forme:

$$q \mapsto exp(q_1 X_1) \circ \cdots \circ exp(q_n X_n) \circ A$$

où A est un élément fixe du groupe G et q décrit un voisinage de 0 dans $\boldsymbol{R}^n$.

A.5.6 Représentation adjointe

Définition A.8

On appelle représentation d'une algèbre de Lie $\mathfrak{g}$ dans une autre algèbre de Lie $\mathfrak{h}$ une application linéaire de $\mathfrak{g}$ dans $\mathfrak{h}$, vérifiant la propriété:

$$f([X,Y]_{\mathfrak{g}}) = [f(X), f(Y)]_{\mathfrak{h}} \qquad \text{(Eq.A.43)}$$

où $[\,,\,]_{\mathfrak{g}}$ et $[\,,\,]_{\mathfrak{h}}$ désignent respectivement les crochets de Lie associés aux structures d'algèbre de Lie des espaces $\mathfrak{g}$ et $\mathfrak{h}$. De plus si la représentation f est invective on dira qu'elle est fidèle.

Soient $\psi : G \to H$ un isomorphisme de groupes de Lie et $\psi^T : \mathfrak{g} \to \mathfrak{h}$ l'isomorphisme d'algèbres de Lie associé. Si en particulier ψ est un automorphisme du groupe de Lie G alors ψ^T est un automorphisme de l'algèbre de Lie $\mathfrak{g}$. L'automorphisme intérieur In_g de G induit l'automorphisme intérieur In_g^T de $\mathfrak{g}$, qu'on notera aussi g_*:

$$g_* = In_g^T;\ \forall g \in G \qquad \text{(Eq.A.44)}$$

Ainsi, il résulte de la relation (Eq.A.44) que:

$$\begin{aligned} exp(g_* X) &= In_g exp(X) \\ &= g\, exp(X)\, g^{-1} \end{aligned} \qquad \text{(Eq.A.45)}$$

d'où:

$$exp(g_* X) = g\, exp(X)\, g^{-1};\ \forall g \in G,\ \forall X \in \mathfrak{g} \qquad \text{(Eq.A.46)}$$

Comme l'application $g \mapsto In_g$ est un homomorphisme de G dans $Aut(G)$ alors:

$$In_g^T \circ In_h^T = In_{g\,h}^T \qquad \text{(Eq.A 47)}$$

et, donc, nous pouvons écrire:

$$g_* \circ h_* = (g\,h)_* \qquad \text{(Eq.A.48)}$$

L'application $(.)_*$ est une action du groupe de Lie G dans son algèbre de Lie $\mathfrak{g}$. Ainsi, pour tout élément $g \in G$, g_* est un automorphisme de l'algèbre de Lie $\mathfrak{g}$. C'est donc une transformation linéaire non singulière de $\mathfrak{g}$ qui satisfait la relation:

$$g_*[X,Y] = [g_* X, g_* Y] \qquad \text{(Eq.A.49)}$$

En résumé, $(.)_*$ est une représentation de G dans $\mathfrak{g}$ appelée représentation adjointe et comme le champ de vecteur est invariant à gauche la relation (Eq.A.19) permet d'écrire:

$$\begin{aligned} g_* Y &= In_g^T Y \\ &= R_{g^{-1}}^T L_g^T Y \\ &= R_{g^{-1}}^T Y \end{aligned} \qquad \text{(Eq.A.50)}$$

ainsi, si X est le générateur infinitésimal du sous-groupe à un paramètre $R_{exp(tX)}$, en posant dans la relation (Eq.A.50) $g = exp(tX)$ il vient:

$$\begin{aligned} (exp\, tX)_* Y &= R_{(exp\ tX)^{-1}}^T Y \\ &= R_{exp(-tX)}^T Y \end{aligned} \qquad \text{(Eq.A.51)}$$

soit en différenciant le terme de droite de la relation (Eq.A.51) relativement à la variable réelle t en $t = 0$ nous obtenons la dérivée de Lie $[X,Y]$, ainsi:

$$\left[\frac{d}{dt}(exp\, tX)_* Y\right]_{t=0} = [X,Y] \qquad \text{(Eq.A.52)}$$

Le groupe $Aut(\mathfrak{g})$ des automorphismes de l'algèbre de Lie $\mathfrak{g}$ dans lui même est un groupe de Lie. Alors, la représentation adjointe $(.)_* : G \to Aut(\mathfrak{g})$ est un homomorphisme de groupes de Lie. L'algèbre de Lie $\mathfrak{a}(\mathfrak{g})$, associée au groupe de Lie $Aut(\mathfrak{g})$, est l'espace de dérivation de $\mathfrak{g}$; c'est à dire, des applications linéaires $D : \mathfrak{g} \to \mathfrak{g}$ telles que:

$$D[Y,Z] = [DY, Z] + [Y, DZ];\ \forall Y, Z \in \mathfrak{g} \qquad \text{(Eq.A.53)}$$

Maintenant soit X un élément quelconque de $\mathfrak{g}$, l'application $Y \mapsto [X,Y]$ définit une dérivation dans $\mathfrak{g}$, puisque d'après l'identité de Jacobi:

$$[X,[Y,Z]] = [[X,Y],Z] + [Y,[X,Z]]$$

D'après la théorie générale, la représentation adjointe $(.)_* : G \to Aut(\mathfrak{g})$ induit un homomorphisme d'algèbres de Lie $(.)_*^T : \mathfrak{g} \to \mathfrak{a}(\mathfrak{g})$ qu'on va noter dans la suite "ad". Pour tout champ de vecteur X de $\mathfrak{g}$, l'homomorphisme ad_X est défini par:

$$ad_X Y = \left[\frac{d}{dt}(exp\, tX)_* Y\right]_{t=0} \qquad \text{(Eq.A.54)}$$

ainsi, compte tenu de la relation (Eq.A.54), on a:

$$ad_X Y = [X,Y] \qquad \text{(Eq.A.55)}$$

Par ailleurs, compte tenu de la relations (Eq.A.55), l'identité de Jacobi devient:

$$ad_X\,[Y,Z] = \; [ad_X\,Y, Z] + [Y, ad_X\,Z];\; \forall Y, Z \in \mathfrak{g} \qquad \text{(Eq.A.56)}$$

ce qui exprime que $ad_X : \mathfrak{g} \mapsto \mathfrak{g}$ peut être interprété comme une dérivée[2] de l'algèbre de Lie $\mathfrak{g}$ [§ (Eq.A.53)]. Reprenons l'identité de Jacobi; compte tenue de la relation (Eq.A.54), elle peut se mettre sous la forme:

$$ad_X \circ ad_Y\; Z = \; ad_{[X,Y]}\; Z + ad_Y \circ ad_X\; Z;\; \forall Z \in \mathfrak{g} \qquad \text{(Eq.A.57)}$$

Finalement, la relation:

$$ad_{[X,Y]} = \; ad_X \circ ad_Y - ad_Y \circ ad_X;\; \forall X, Y \in \mathfrak{g} \qquad \text{(Eq.A.58)}$$

La relation (Eq.A.58) représente une interprétation de l'identité de Jacobi pour le crochet de Lie [§ (Eq.A.23)] en termes d'endomorphismes de l'algèbre de Lie $\mathfrak{g}$. Elle permet ainsi le calcul de la dérivée de Lie associée au produit du crochet de Lie de deux champs de vecteurs en fonction des dérivées de Lie qui leurs sont associées.

[2]Elle est dite dérivée de Lie.

Annexe B

MODELE GEOMETRIQUE DU ROBOT ACMA–H80

B.1 Description du robot ACMA–H80

Le robot ACMA–H80 se compose d'un porteur et d'un poignet. La chaîne cinématique du porteur du robot est du type *PRR* et son poignet comporte trois rotations d'axes concourants (voir Fig. B.1). Tous les calculs nécessaires au traitement de cet exemple ont été effectués à l'aide de notre logiciel[1] `dualstruc.map`:

– Paramètres de Denavit–Hartenberg du robot ACMA–H80 –

Elément d'indice i	γ_i	l_i	θ_i	d_i
1	0	0	0	d_1
2	0	0	θ_2	0
3	0	L_3	θ_3	0
4	0	L_4	θ_4	0
5	$-\pi/2$	0	θ_5	0
6	$-\pi/2$	0	θ_6	0

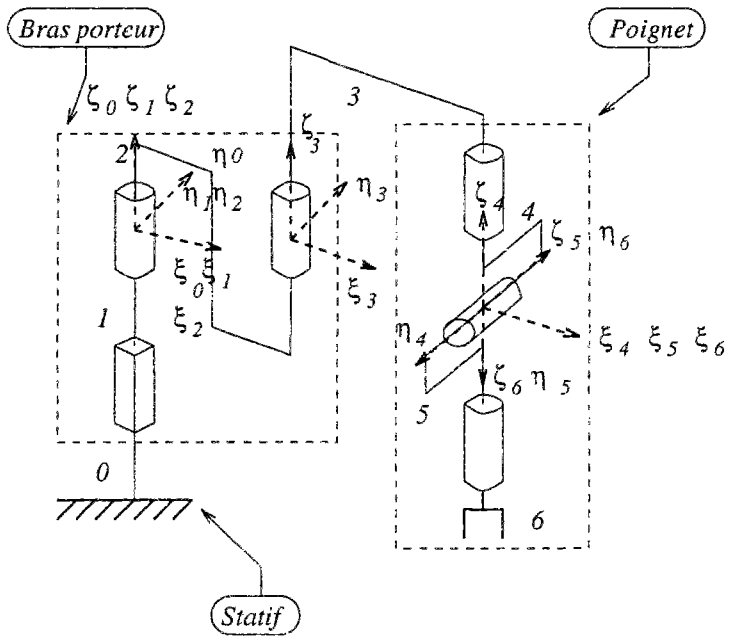
B.2 Modèle géométrique

Notons q_i la $i^{ém}$ coordonnée articulaire. Ainsi, nous avons le renommage de variables suivant:

– Paramètres articulaires du robot ACMA–H80 –

Elément d'indice i	*Type de l'articulation*	*Paramètre de D–H*	*Variable articulaire*
1	P	d_1	q_1
2	R	θ_2	q_2
3	R	θ_3	q_3
4	R	θ_4	q_4
5	R	θ_5	q_5
6	R	θ_6	q_6

[1] Voir le chapitre 7 [§ Page 195].



Cette figure schématise les articulations du robot ACMA-H80 et les familles fondamentales associées aux éléments de la décomposition du robot avec les conventions de Denavit-Hartenberg.

Fig. B.1: Schématisation du robot ACMA-H80

Les matrices duales de passages, entre les éléments adjacents du robot, associées aux convenions de Denavit-Hartenberg sont:

- pour le porteur:

$$[\boldsymbol{D}^1_{0*}] = \begin{pmatrix} 1 & -\varepsilon\, q_1 & 0 \\ \varepsilon\, q_1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$[\boldsymbol{D}^2_{1*}] = \begin{pmatrix} \cos(q_2) & -\sin(q_2) & 0 \\ \sin(q_2) & \cos(q_2) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$[\boldsymbol{D}^3_{2*}] = \begin{pmatrix} \cos(q_3) & -\sin(q_3) & 0 \\ \sin(q_3) & \cos(q_3) & -\varepsilon\, L_3 \\ \varepsilon\, L_3\, \sin(q_3) & \varepsilon\, L_3\, \cos(q_3) & 1 \end{pmatrix}$$

- pour le poignet:

$$[\boldsymbol{D}_{3*}^{4}] = \begin{pmatrix} \cos(q_4) & -\sin(q_4) & 0 \\ \sin(q_4) & \cos(q_4) & -\varepsilon L_4 \\ \varepsilon L_4 \sin(q_4) & \varepsilon L_4 \cos(q_4) & 1 \end{pmatrix}$$

$$[\boldsymbol{D}_{4*}^{5}] = \begin{pmatrix} \cos(q_5) & -\sin(q_5) & 0 \\ 0 & 0 & 1 \\ -\sin(q_5) & -\cos(q_5) & 0 \end{pmatrix}$$

$$[\boldsymbol{D}_{5*}^{6}] = \begin{pmatrix} \cos(q_6) & -\sin(q_6) & 0 \\ 0 & 0 & 1 \\ -\sin(q_6) & -\cos(q_6) & 0 \end{pmatrix}$$

Le déplacement de passage qui caractérise la configuration de l'organe terminal est donnée par le produit de ces six déplacements:

$$\boldsymbol{D}_0^6 = \boldsymbol{D}_5^6\, \boldsymbol{D}_4^5\, \boldsymbol{D}_3^4\, \boldsymbol{D}_2^3\, \boldsymbol{D}_1^2\, \boldsymbol{D}_0^1$$

Ainsi, le modèle géométrique du robot, qui en découle est donnée par la matrice duale $[\boldsymbol{D}_{0*}^{6}]$. Après regroupement des paramètres angulaire, on obtient, les composantes duale de cette matrice:

$$[\boldsymbol{D}_{0*}^{6}]_{1,1} = \left(-\frac{q_1 \sin(q_6+q_5+q_4+q_3+q_2)}{2} - \frac{q_1 \sin(-q_6+q_5+q_4+q_3+q_2)}{2} + \frac{L_3 \cos(q_6+q_3+q_2)}{2} - \frac{L_3 \cos(-q_6+q_3+q_2)}{2} + \frac{L_4 \cos(q_6+q_4+q_3+q_2)}{2} - \frac{L_4 \cos(-q_6+q_4+q_3+q_2)}{2}\right)\varepsilon + \frac{\cos(q_6+q_5+q_4+q_3+q_2)}{2} + \frac{\cos(-q_6+q_5+q_4+q_3+q_2)}{2}$$

$$[\boldsymbol{D}_{0*}^{6}]_{1,2} = \left(\frac{L_3 \sin(-q_6+q_3+q_2)}{2} - \frac{L_4 \sin(q_6+q_4+q_3+q_2)}{2} - \frac{q_1 \cos(-q_6+q_5+q_4+q_3+q_2)}{2} - \frac{q_1 \cos(q_6+q_5+q_4+q_3+q_2)}{2} - \frac{L_3 \sin(q_6+q_3+q_2)}{2} + \frac{L_4 \sin(-q_6+q_4+q_3+q_2)}{2}\right)\varepsilon - \frac{\sin(q_6+q_5+q_4+q_3+q_2)}{2} - \frac{\sin(-q_6+q_5+q_4+q_3+q_2)}{2}$$

$$[\boldsymbol{D}_{0*}^{6}]_{1,3} = \left(\frac{L_3 \sin(q_6+q_5+q_4)}{2} + \frac{L_3 \sin(-q_6+q_5+q_4)}{2} + \frac{L_4 \sin(q_6+q_5)}{2} + \frac{L_4 \sin(-q_6+q_5)}{2}\right)\varepsilon - \sin(q_6)$$

$$[\boldsymbol{D}_{0*}^{6}]_{2,1} = -\sin(q_5+q_4+q_3+q_2) - \varepsilon\, q_1 \cos(q_5+q_4+q_3+q_2)$$

$$[\boldsymbol{D}_{0*}^{6}]_{2,2} = -\cos(q_5+q_4+q_3+q_2) + \varepsilon\, q_1 \sin(q_5+q_4+q_3+q_2)$$

$$[\boldsymbol{D}_{0*}^{6}]_{2,3} = (L_3 \cos(q_5+q_4) + \cos(q_5)\, L_4)\,\varepsilon$$

$$[\boldsymbol{D}_{0*}^{6}]_{3,1} = \left(-\frac{L_3 \sin(-q_6+q_3+q_2)}{2} - \frac{L_4 \sin(-q_6+q_4+q_3+q_2)}{2} - \frac{L_3 \sin(q_6+q_3+q_2)}{2} - \frac{q_1 \cos(q_6+q_5+q_4+q_3+q_2)}{2} + \frac{q_1 \cos(-q_6+q_5+q_4+q_3+q_2)}{2} - \frac{L_4 \sin(q_6+q_4+q_3+q_2)}{2}\right)\varepsilon + \frac{\sin(-q_6+q_5+q_4+q_3+q_2)}{2} - \frac{\sin(q_6+q_5+q_4+q_3+q_2)}{2}$$

$$[\boldsymbol{D}_{0*}^{6}]_{3,2} = \left(-\frac{L_3 \cos(q_6+q_3+q_2)}{2} - \frac{L_4 \cos(q_6+q_4+q_3+q_2)}{2} - \frac{L_4 \cos(-q_6+q_4+q_3+q_2)}{2} - \frac{L_3 \cos(-q_6+q_3+q_2)}{2} - \frac{q_1 \sin(-q_6+q_5+q_4+q_3+q_2)}{2} + \frac{q_1 \sin(q_6+q_5+q_4+q_3+q_2)}{2}\right)\varepsilon - \frac{\cos(q_6+q_5+q_4+q_3+q_2)}{2} + \frac{\cos(-q_6+q_5+q_4+q_3+q_2)}{2}$$

$$[\boldsymbol{D}_{0*}^{6}]_{3,3} = \left(\frac{L_3 \cos(q_6+q_5+q_4)}{2} - \frac{L_3 \cos(-q_6+q_5+q_4)}{2} + \frac{L_4 \cos(q_6+q_5)}{2} - \frac{L_4 \cos(-q_6+q_5)}{2}\right)\varepsilon - \cos(q_6)$$

B.3 Configuration de l'organe terminal dans l'espace opérationnel

Dans l'application suivante nous cherchons à exprimer les coordonnées opérationnelles en fonction des coordonnées articulaires du robot ACMA-H80. En d'autre termes, on cherche à exprimer le modèle géométrique inverse du robot. Plusieurs paramétrages du groupe des déplacements euclidiens sont possibles. Nous donnerons ici le modèle géométrique inverse associé au paramétrage de $\mathbb{D}$ par ce que nous avons convenu d'appeler "les angles d'Euler duaux".

B.3.1 Calcul formel

D'après les relations (Eq.3.39), les angles d'Euler duaux associés à la transformation $[\boldsymbol{D}^6_{0*}]$ sont:

$$\begin{cases} \theta = arctg\left(\dfrac{\sqrt{[\boldsymbol{D}^6_{0*}]^2_{3,1} + [\boldsymbol{D}^6_{0*}]^2_{3,2}}}{[\boldsymbol{D}^6_{0*}]_{3,3}}\right) \\ \phi = arctg\left(\dfrac{[\boldsymbol{D}^6_{0*}]_{2,3}/sin\,\theta}{[\boldsymbol{D}^6_{0*}]_{1,3}/sin\,\theta}\right) \\ \psi = arctg\left(\dfrac{[\boldsymbol{D}^6_{0*}]_{3,2}/sin\,\theta}{-[\boldsymbol{D}^6_{0*}]_{3,1}/sin\,\theta}\right) \end{cases}$$

Ainsi, le calcul et la simplification symbolique de ces trois paramètres duaux grâce aux outils définis dans le module `dualstruc.map` donne les expressions algébriques suivantes:

- pour θ:

```
                1/2
theta := (I 2    (- sin(%2) sin(%1) L4 cos(- q[6] + q[5])

    + cos(%2) cos(%1) L3 cos(q[6] + q[5] + q[4])

    + sin(%2) sin(%1) L3 cos(q[6] + q[5] + q[4])

    + L3 cos(%4) cos(%2) cos(q[6]) - L4 sin(%5) sin(%1) cos(q[6])

    + L4 sin(%7) sin(%2) cos(q[6]) - L3 sin(%6) sin(%1) cos(q[6])

    + L3 cos(%6) cos(%2) cos(q[6]) - L4 sin(%7) sin(%1) cos(q[6])

    + L3 sin(%4) sin(%2) cos(q[6]) + L4 sin(%5) sin(%2) cos(q[6])

    - L3 sin(%4) sin(%1) cos(q[6]) - L3 cos(%6) cos(%1) cos(q[6])

    - L4 cos(%7) cos(%1) cos(q[6]) + L4 cos(%7) cos(%2) cos(q[6])

    - L4 cos(%5) cos(%1) cos(q[6]) + L3 sin(%6) sin(%2) cos(q[6])

    + L4 cos(%5) cos(%2) cos(q[6]) - L3 cos(%4) cos(%1) cos(q[6])

    - cos(%2) cos(%1) L3 cos(- q[6] + q[5] + q[4])

    + cos(%2) cos(%1) L4 cos(q[6] + q[5])
```

```
      - cos(%2) cos(%1) L4 cos(- q[6] + q[5])

      - sin(%2) sin(%1) L3 cos(- q[6] + q[5] + q[4])

      + sin(%2) sin(%1) L4 cos(q[6] + q[5]) - L3 cos(q[6] + q[5] + q[4])

      + L3 cos(- q[6] + q[5] + q[4]) + L4 cos(- q[6] + q[5])

                                                  1/2   1/2
                                                 2    %3       1/2
      - L4 cos(q[6] + q[5])) eps - 2 I arctanh(1/2 ----------) %3
                                                   cos(q[6])

                  2                                                 /
     (- 2 cos(q[6])  + cos(%2) cos(%1) - 1 + sin(%2) sin(%1))  /  (
                                                              /

          1/2          2       1/2       1/2
     - 4 %3    cos(q[6])  - 2 %3    + 2 %3    sin(%2) sin(%1)

           1/2
      + 2 %3    cos(%2) cos(%1))

%1 :=                q[6] + q[5] + q[4] + q[3] + q[2]

%2 :=               - q[6] + q[5] + q[4] + q[3] + q[2]

%3 :=              cos(%2) cos(%1) - 1 + sin(%2) sin(%1)

%4 :=                      q[6] + q[3] + q[2]

%5 :=                    - q[6] + q[4] + q[3] + q[2]

%6 :=                      - q[6] + q[3] + q[2]

%7 :=                     q[6] + q[4] + q[3] + q[2]
```

où le symbole `arctanh` désigne la fonction arc tangente hyperbolique et `I` l'unité complexe i telle que $i^2 = -1$.

- pour ϕ:

```
phi := ((8 q[1] - 2 cos(%2) L3 sin(%5) + 2 cos(%1) L4 sin(%4)

     - 4 cos(%2) q[1] cos(%1) - 2 cos(%2) L3 sin(%6) + 2 cos(%1) L3 sin(%6)

     + 2 cos(%1) L3 sin(%5) + 2 cos(%1) L4 sin(%7)

     - 2 cos(%1) L4 sin(%4) sin(%2) sin(%1)

     - 2 cos(%1) L4 sin(%7) sin(%2) sin(%1)

     - 2 cos(%1) L3 sin(%5) sin(%2) sin(%1)

     - 2 cos(%1) L3 sin(%6) sin(%2) sin(%1)
```

```
+ 2 cos(%2) L3 sin(%5) sin(%2) sin(%1)

+ 2 cos(%2) L4 sin(%4) sin(%2) sin(%1)

+ 2 cos(%2) L4 sin(%7) sin(%2) sin(%1)

+ 2 cos(%2) L3 sin(%6) sin(%2) sin(%1)

+ 4 cos(%2) q[1] cos(%1) sin(%2) sin(%1) - 4 L4 cos(%7) sin(%1)

- 4 L4 cos(%4) sin(%1) + 4 L3 cos(%5) sin(%2)

                              2
- cos(%1) L4 sin(%4) cos(%2)  + 4 L4 cos(%7) sin(%2)

- 2 cos(%2) L4 sin(%7) + 4 L4 cos(%4) sin(%2) - 4 L3 cos(%6) sin(%1)

+ 4 L3 cos(%6) sin(%2) - 2 cos(%2) L4 sin(%4)

                             2
+ L4 cos(%7) sin(%1) cos(%1)  - 8 q[1] sin(%2) sin(%1)

                                                    2
- 4 L3 cos(%5) sin(%1) - L3 cos(%6) sin(%2) cos(%2)

                                 2                                  2
+ 2 q[1] sin(%2) sin(%1) cos(%1)  + 2 q[1] sin(%1) sin(%2) cos(%2)

                              2                 2        2
+ L3 cos(%5) sin(%1) cos(%1)  + 4 q[1] cos(%2)  cos(%1)

                              2                                2
- L3 cos(%5) sin(%2) cos(%2)  - cos(%1) L4 sin(%7) cos(%2)

                              2          3                     3
+ L3 cos(%6) sin(%1) cos(%1)  - cos(%1)  L4 sin(%4) - cos(%1)  L4 sin(%7)

                                2                                  2
- 3 L4 cos(%4) sin(%2) cos(%1)  + 3 L3 cos(%5) sin(%1) cos(%2)

                              2                                2
- L4 cos(%4) sin(%2) cos(%2)  + L4 cos(%4) sin(%1) cos(%1)

                              2          3
- L4 cos(%7) sin(%2) cos(%2)  + cos(%2)  L4 sin(%4)

                                2          3
- 3 L4 cos(%7) sin(%2) cos(%1)  + cos(%2)  L3 sin(%5)

                              2                                  2
- cos(%1) L3 sin(%6) cos(%2)  + 3 L4 cos(%7) sin(%1) cos(%2)

         3                                          2
- cos(%1)  L3 sin(%5) - cos(%1) L3 sin(%5) cos(%2)
```

```
                                    2          3
      + cos(%2) L3 sin(%6) cos(%1)  + cos(%2)  L3 sin(%6)

                3                                       3
      + 2 cos(%2)  q[1] cos(%1) + 2 cos(%2) q[1] cos(%1)

                                    2                                  2
      + cos(%2) L4 sin(%7) cos(%1)  + cos(%2) L3 sin(%5) cos(%1)

              3                       3
      - cos(%1)  L3 sin(%6) + cos(%2)  L4 sin(%7)

                                    2                                  2
      + 3 L3 cos(%6) sin(%1) cos(%2)  + cos(%2) L4 sin(%4) cos(%1)

                                    2                                  2
      - 3 L3 cos(%6) sin(%2) cos(%1)  - 3 L3 cos(%5) sin(%2) cos(%1)

                                    2                 2                2
      + 3 L4 cos(%4) sin(%1) cos(%2)  - 6 q[1] cos(%1)  - 6 q[1] cos(%2) ) eps

                  2        2              3                          3
      - 4 %3 cos(%2)  cos(%1)  - 2 %3 cos(%2)  cos(%1) - 2 %3 cos(%1)  cos(%2)

      + 4 %3 cos(%2) cos(%1) + 8 %3 sin(%2) sin(%1)

                                                                  2
      - 4 %3 sin(%2) sin(%1) cos(%2) cos(%1) - 2 %3 cos(%2)  sin(%2) sin(%1)

                                    2                2              2
      - 2 %3 sin(%2) sin(%1) cos(%1)  + 6 %3 cos(%2)  + 6 %3 cos(%1)  - 8 %3)

        /                 2        2            3                     3
       /  (8 + 4 cos(%2)  cos(%1)  + 2 cos(%2)  cos(%1) + 2 cos(%1)  cos(%2)
      /

                                                                 2           2
      + 4 sin(%2) sin(%1) cos(%2) cos(%1) - 6 cos(%2)  - 6 cos(%1)

                                                                    2
      - 4 cos(%2) cos(%1) - 8 sin(%2) sin(%1) + 2 cos(%2)  sin(%2) sin(%1)

                                    2
      + 2 sin(%2) sin(%1) cos(%1) )

%1 :=                 q[6] + q[5] + q[4] + q[3] + q[2]

%2 :=                - q[6] + q[5] + q[4] + q[3] + q[2]

                                cos(%2) - cos(%1)
%3 :=                    arctan(----------------)
                                sin(%2) - sin(%1)

%4 :=                     - q[6] + q[4] + q[3] + q[2]

%5 :=                       - q[6] + q[3] + q[2]
```

```
%6 :=                    q[6] + q[3] + q[2]

%7 :=                 q[6] + q[4] + q[3] + q[2]
```

- pour ψ:

```
                    (L3 cos(q[5] + q[4]) + cos(q[5]) L4) eps
psi := - ---------------------------------------
                                sin(q[6])
```

B.3.2 Application numérique

Maintenant, si l'on suppose que les paramètres constants L_3 et L_4 sont égaux ($L_3 = L_4 = 50\,cm$) et si l'on se donne la configuration articulaire suivante:

– *Configuration articulaire du robot ACMA–H80* –			
Elément d'indice i	*Type de l'articulation*	*Variable articulaire*	*Valeur*
1	P	q_1	$10\,cm$
2	R	q_2	$\pi/4$
3	R	q_3	$\pi/4$
4	R	q_4	$\pi/4$
5	R	q_5	$\pi/4$
6	R	q_6	$\pi/4$

les coordonnées duales correspondant à la configuration de l'organe terminal sont données par le tableau suivant:

– *Configuration de l'organe terminal du robot ACMA–H80* –	
Angles d'Euler duaux	*Angle dual en (radian,cm)*
Précession duale: ϕ	$\left(10 - 25\sqrt{2}\right)\,\varepsilon$
Nutation duale: θ	$-\frac{\pi}{4} + \frac{\left(50\sqrt{2} + 100\right)}{2}\,\varepsilon$
Rot. prop. duale: ψ	$-50\,\varepsilon$

En séparant les parties réelles et duales des angles duaux on obtient les six coordonnées articulaires réelles.

– *Coordonnées opérationnelles du robot ACMA–H80* –		
Axe de transformation	*Angles d'Euler classiques en radian*	*Translations cardans en cm*
Λ_ζ	$\alpha = 0\ radian$	$a = \left(10 - 25\sqrt{2}\right)\ cm$
Λ_η	$\beta = -\frac{\pi}{4}\ radian$	$b = \frac{\left(50\sqrt{2} + 100\right)}{2}\ cm$
Λ_ζ	$\gamma = 0\ radian$	$c = -50\ cm$

Annexe C

SCHEMAS OPTIMAUX DU CALCUL DES ANGLES D'EULER ET DE BRIANT DUAUX

C.1 Introduction

Dans cet annexe nous proposons des schémas optimaux pour le calcul et la simulation numérique de ce que nous avons convenu d'appeler les angles d'Euler et de Briant duaux associés à un déplacement donnée.

C.2 Position du problème

Dans les schémas que nous proposons dans les sections C.3 et C.4, on suppose que la matrice duale associée à la représentation adjointe d'un déplacement $\boldsymbol{A}$ est donnée relativement à une famille fondamentale $\mathfrak{f}$:

$$[\boldsymbol{A}_*] = \begin{pmatrix} rA_{1,1} + \varepsilon\, dA_{1,1} & rA_{1,2} + \varepsilon\, dA_{1,2} & rA_{1,3} + \varepsilon\, dA_{1,3} \\ rA_{2,1} + \varepsilon\, dA_{2,1} & rA_{2,2} + \varepsilon\, dA_{2,2} & rA_{2,3} + \varepsilon\, dA_{2,3} \\ rA_{3,1} + \varepsilon\, dA_{3,1} & rA_{3,2} + \varepsilon\, dA_{3,2} & rA_{3,3} + \varepsilon\, dA_{3,3} \end{pmatrix}$$

où les paramètres symboliques $rA_{i,j}$ et $dA_{i,j}$ désignent respectivement les parties réelle et duale de la composante $[\boldsymbol{A}_*]_{i,j}$.

L'implémentation automatique des schémas de calcul des paramètres associés aux parties réelle et duale des angles d'Euler et de Briant duaux a été décrite grâce aux outils définis dans les modules `dualstruc.map` et `defstruc.map` que nous avons écrit pour Maple V.

C.3 Schéma de calcul des paramètres associés aux angles d'Euler duaux

L'implémentation du schéma optimal de calcul des angles d'Euler duaux suppose que la matrice $[\boldsymbol{A}_*]$ est Δ-orthogonale et qu'elle vérifie les deux hypothèses suivantes:

$$\begin{cases} rA_{1,3} \neq 0 \\ rA_{3,1}^2 + rA_{3,2}^2 \neq 0 \end{cases}$$

```
c     ------------------------------------------------------
c     calcul des parametres associes aux angles d'Euler duaux
c     ------------------------------------------------------
c
c     calcul de la partie reelle de theta
c
      crA31 = rA(3,1)**2
      crA32 = rA(3,2)**2
      scrA = crA31+crA32
      crA = sqrt(scrA)
      beta = atan(1/rA(3,3)*crA)
c
c     calcul de la partie duale de theta
c
      crA33 = rA(3,3)**2
      b = -(crA31*dA(3,3)+crA32*dA(3,3)-dA(3,1)*rA(3,1)*rA(3,3
     $ )-dA(3,2)*rA(3,2)*rA(3,3))/crA/(crA33+scrA)
c
c     calcul de la partie reelle de psi
c
      gamma = atan(rA(2,3)/rA(1,3))
c
c     calcul de la partie duale de psi
c
      crA13 = rA(1,3)**2
      crA23 = rA(2,3)**2
      c = (-rA(2,3)*dA(1,3)+dA(2,3)*rA(1,3))/(crA13+crA23)
c
c     calcul de la partie reelle de phi
c
      alpha = -atan(rA(3,2)/rA(1,3))
c
c     calcul de la partie duale de phi
c
      a = (rA(3,2)*dA(1,3)-dA(3,2)*rA(1,3))/(crA13+crA32)
c
c     ------------------------------------------------------
```

C.4 Schéma de calcul des paramètres associés aux angles de Briant duaux

Comme pour le schéma précedent, l'implémentation du schéma optimal de calcul des angles de Briant duaux suppose que la matrice $[\boldsymbol{A}_*]$ est orthogonale et qu'elle vérifie deux hypothèses:

$$\begin{cases} rA_{3,3} \neq 0 \\ rA_{1,1}^2 + rA_{2,1}^2 \neq 0 \end{cases}$$

```
c     ------------------------------------------------------
c     calcul des parametres associes aux angles de Briant duaux
c     ------------------------------------------------------
c
c     calcul de la partie reelle de theta
c
      crA11 = rA(1,1)**2
      crA21 = rA(2,1)**2
      crA = sqrt(crA11+crA21)
      beta = -atan(rA(3,1)/crA)
c
c     calcul de la partie duale de theta
c
      scrA = crA+crA21
      sqrA = sqrt(scrA)
      ssrA = scrA**2
      crA31 = rA(3,1)**2
      b = -(-rA(3,1)*dA(1,1)*rA(1,1)-rA(3,1)*dA(2,1)*rA(2,1)+
     $ dA(3,1)*crA11+dA(3,1)*crA21)*sqrA/ssrA/(1+crA31/scrA)
c
c     calcul de la partie reelle de psi
c
      gamma = atan(rA(2,1)/rA(1,1))
c
c     calcul de la partie duale de psi
c
      c = (-rA(2,1)*dA(1,1)+dA(2,1)*rA(1,1))/(crA11+crA21)
c
c     calcul de la partie reelle de phi
c
      alpha = atan(rA(3,2)/rA(3,3))
c
c     calcul de la partie duale de phi
c
      crA33 = rA(3,3)**2
      crA32 = rA(3,2)**2
      t10 = -(rA(3,2)*dA(3,3)-dA(3,2)*rA(3,3))/(crA33+crA32)
c
c     ------------------------------------------------------
```

C.5 Application

Revenons sur l'exemple du modèle géométrique inverse du robot ACMA–H80. Rappelons que le bras porteur du robot est une chaîne cinématique de type *PRR* et que son poignet est de type *RRR* à axes concourants. Les paramètres de Denavit-Hartenberg du robot ACMA–H80 étant:

– Paramètres de Denavit–Hartenberg du robot ACMA–H80 –				
Elément d'indice i	γ_i	l_i	θ_i	d_i
1	0	0	0	d_1
2	0	0	θ_2	0
3	0	L_3	θ_3	0
4	0	L_4	θ_4	0
5	$-\pi/2$	0	θ_5	0
6	$-\pi/2$	0	θ_6	0

comme précédement, en notant q_i la $i^{ém}$ coordonnée articulaire, nous avons le renommage de variables suivant:

– Paramètres articulaires du robot ACMA–H80 –			
Elément d'indice i	Type de l'articulation	Paramètre de D–H	Variable articulaire
1	P	d_1	q_1
2	R	θ_2	q_2
3	R	θ_3	q_3
4	R	θ_4	q_4
5	R	θ_5	q_5
6	R	θ_6	q_6

Nous l'avons déjà vu, la matrice duale qui décrit la configuration de l'organe terminale peut être calculée à partire de ses composantes duales:

$$[\boldsymbol{D}^6_{0*}]_{1,1} = \left(-\frac{q_3\,\sin(q_6+q_5+q_4+q_3+q_2)}{2} - \frac{q_1\,\sin(-q_6+q_5+q_4+q_3+q_2)}{2} + \frac{L_3\,\cos(q_6+q_3+q_2)}{2} - \frac{L_3\,\cos(-q_6+q_5+q_2)}{2} + \frac{L_4\,\cos(q_6+q_4+q_3+q_2)}{2} - \frac{L_4\,\cos(-q_6+q_4+q_3+q_2)}{2}\right)\varepsilon + \frac{\cos(q_6+q_5+q_4+q_3+q_2)}{2} + \frac{\cos(-q_6+q_5+q_4+q_3+q_2)}{2}$$

$$[\boldsymbol{D}^6_{0*}]_{1,2} = \left(\frac{L_3\,\sin(-q_6+q_3+q_2)}{2} - \frac{L_4\,\sin(q_6+q_4+q_3+q_2)}{2} - \frac{q_1\,\cos(-q_6+q_5+q_4+q_3+q_2)}{2} - \frac{q_1\,\cos(q_6+q_5+q_4+q_3+q_2)}{2} - \frac{L_3\,\sin(q_6+q_3+q_2)}{2} + \frac{L_4\,\sin(-q_6+q_4+q_3+q_2)}{2}\right)\varepsilon - \frac{\sin(q_6+q_5+q_4+q_3+q_2)}{2} - \frac{\sin(-q_6+q_5+q_4+q_3+q_2)}{2}$$

$$[\boldsymbol{D}^6_{0*}]_{1,3} = \left(\frac{L_3\,\sin(q_6+q_5+q_4)}{2} + \frac{L_3\,\sin(-q_6+q_5+q_4)}{2} + \frac{L_4\,\sin(q_6+q_5)}{2} + \frac{L_4\,\sin(-q_6+q_5)}{2}\right)\varepsilon - \sin(q_6)$$

$$[\boldsymbol{D}^6_{0*}]_{2,1} = -\sin(q_5+q_4+q_3+q_2) - \varepsilon\, q_1\,\cos(q_5+q_4+q_3+q_2)$$

$$[\boldsymbol{D}^6_{0*}]_{2,2} = -\cos(q_5+q_4+q_3+q_2) + \varepsilon\, q_1\,\sin(q_5+q_4+q_3+q_2)$$

$$[\boldsymbol{D}^6_{0*}]_{2,3} = (L_3\,\cos(q_5+q_4) + \cos(q_5)\,L_4)\,\varepsilon$$

$$[\boldsymbol{D}^6_{0*}]_{3,1} = \left(-\frac{L_3\sin(-q_6+q_3+q_2)}{2} - \frac{L_4\sin(-q_6+q_4+q_3+q_2)}{2} - \frac{L_3\sin(q_6+q_3+q_2)}{2} - \frac{q_1\cos(q_6+q_5+q_4+q_3+q_2)}{2} + \frac{q_1\cos(-q_6+q_5+q_4+q_3+q_2)}{2} - \frac{L_4\sin(q_6+q_4+q_3+q_2)}{2}\right)\varepsilon + \frac{\sin(-q_6+q_5+q_4+q_3+q_2)}{2} - \frac{\sin(q_6+q_5+q_4+q_3+q_2)}{2}$$

$$[\boldsymbol{D}^6_{0*}]_{3,2} = \left(-\frac{L_3\cos(q_6+q_3+q_2)}{2} - \frac{L_4\cos(q_6+q_4+q_3+q_2)}{2} - \frac{L_4\cos(-q_6+q_4+q_3+q_2)}{2} - \frac{L_3\cos(-q_5+q_3+q_2)}{2} - \frac{q_1\sin(-q_6+q_5+q_4+q_3+q_2)}{2} + \frac{q_1\sin(q_6+q_5+q_4+q_3+q_2)}{2}\right)\varepsilon - \frac{\cos(q_6+q_5+q_4+q_3+q_2)}{2} + \frac{\cos(-q_5+q_5+q_4+q_3+q_2)}{2}$$

$$[\boldsymbol{D}^6_{0*}]_{3,3} = \left(\frac{L_3\cos(q_6+q_5+q_4)}{2} - \frac{L_3\cos(-q_6+q_5+q_4)}{2} + \frac{L_4\cos(q_6+q_5)}{2} - \frac{L_4\cos(-q_6+q_5)}{2}\right)\varepsilon - \cos(q_6)$$

Le robot ACMA–H80, comme tout mécanisme est une chaîne cinématique à mouvement forcé. Les mouvements des éléments qui le composent sont établits relativement à un élément de référence. Ces mouvements peuvent être de natures diverses. Les trajectoires suivent donc des loi de différents types: linéaire, Bang-Bang, polynomiale -en général, dans les problèmes d'ordre pratique, de degrés trois ou cinq- ... etc. A titre d'application, on s'interesse à savoire comment varient les six paramètres rééls -decrivants l'espace opérationnel- associés aux angles d'Euler et de Briant duaux. Ainsi nous proposons d'étudier leurs variations dans le cas où toutes les trjectoires suivent une loi Bang-Bang.

C.5.1 Description des trajectoires – Loi Bang-Bang

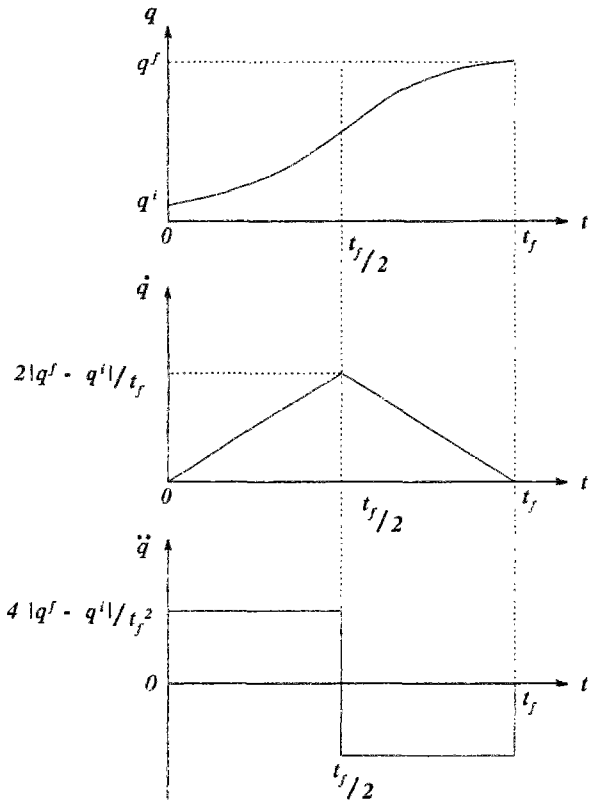
Dans des problèmes de simulation, on est souvent amener à préciser les lois que suivent les trajectoires. Une trajectoire qui suit une loi Bang-Bang comporte une phase accélérée j'usqu'à l'instant $t_f/2$ et une phase décélérée. Dans ces deux phases, le mouvement s'effectue avec des accélérations constantes telles que les vitesses de départ et d'arrivée soient nulles. Ainsi, le mouvement est continu en position et en vitesse et discontinu en accélération [Fig. C.1]. La position articulaire correspondant à un tel mouvement est donnée par:

$$\begin{cases} q(t) = 2\,(q^f - q^i)\left(\dfrac{t}{t_f}\right)^2 + q^i & \text{si } 0 \le t \le \dfrac{t_f}{2} \\ q(t) = -2\,(q^f - q^i)\left[\left(\dfrac{t}{t_f}\right)^2 - 2\,\dfrac{t}{t_f} + \dfrac{1}{2}\right] + q^i & \text{si } \dfrac{t_f}{2} \le t \le t_f \end{cases}$$

Ainsi, dans la suite, toutes les trajectoires seront supposées suivre une loi Bang-Bang.

C.5.2 Simulation des configurations de l'effecteur du robot ACMA–H80

Cette simulation, sera faite à partir des angles d'Euler et de Briant duaux. On prendra le centimètre comme unité de mesure des distance et le radian pour la mesure des angles.



Le mouvement est continu en position et en vitesse mais discontinu en accélération. Les accélérations durant ces deux phases sont égales en valeurs absolues et de signes contraires.

Fig. C.1: Loi Bang-Bang

– *Configurations articulaires du robot ACMA–H80* –			
Elément d'indice j	*Type de l'articulation*	*Configuration initiale* q_j^i	*Configuration finale* q_j^f
1	P	0 *cm*	10 *cm*
2	R	0 *radian*	10 *radian*
3	R	0 *radian*	10 *radian*
4	R	0 *radian*	10 *radian*
5	R	0 *radian*	10 *radian*
6	R	0 *radian*	10 *radian*

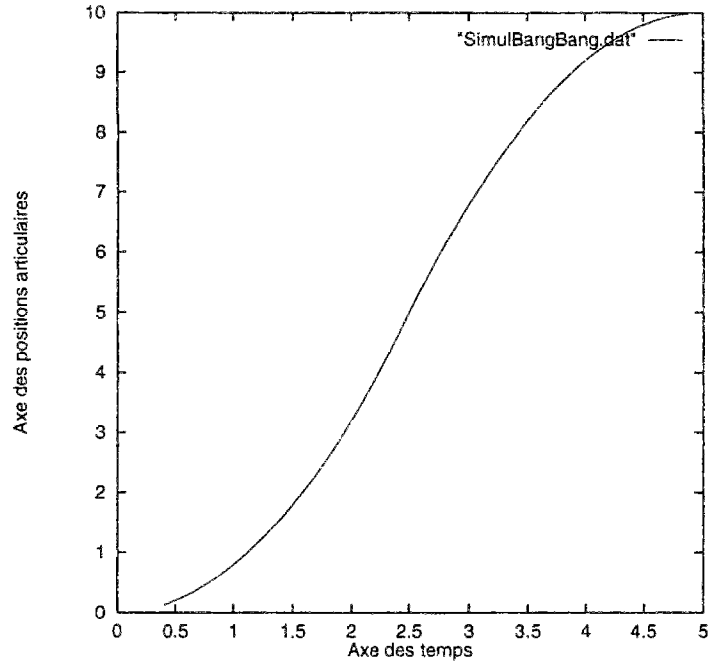


Fig. C.2: *Simulation numérique de la loi Bang-Bang*

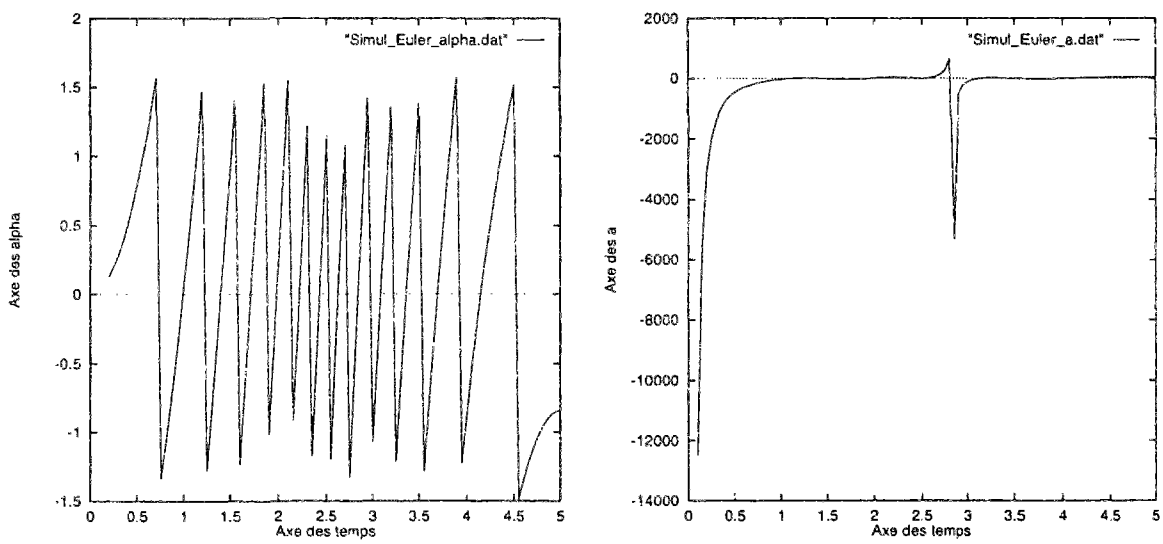


Fig. C.3: *Simulation numérique de la précession duale de l'organe terminal du robot ACMA–H80*

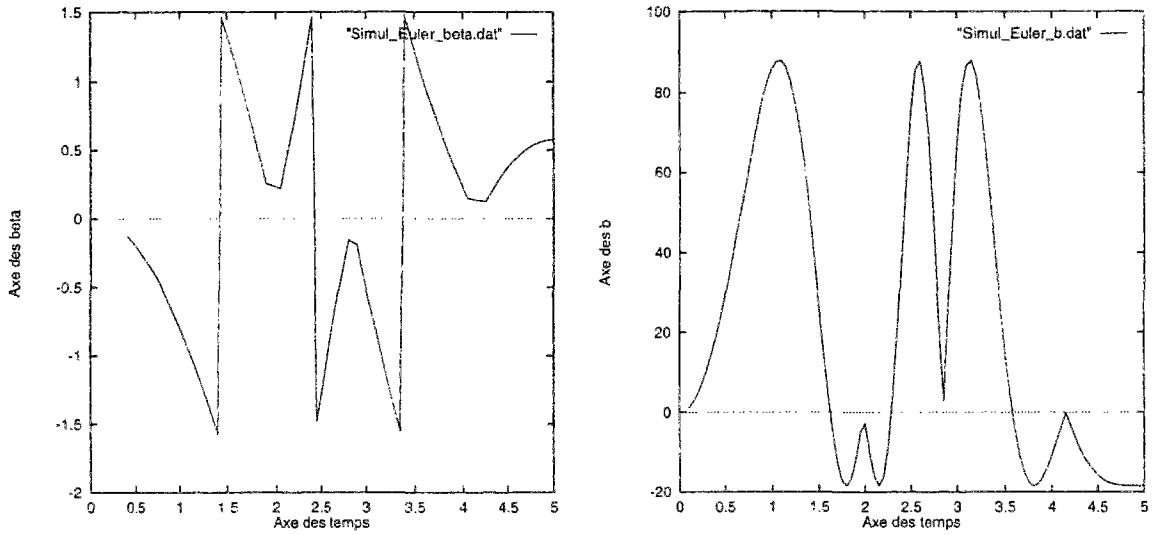


Fig. C.4: *Simulation numérique de la nutation duale de l'organe terminal du robot ACMA–H80*

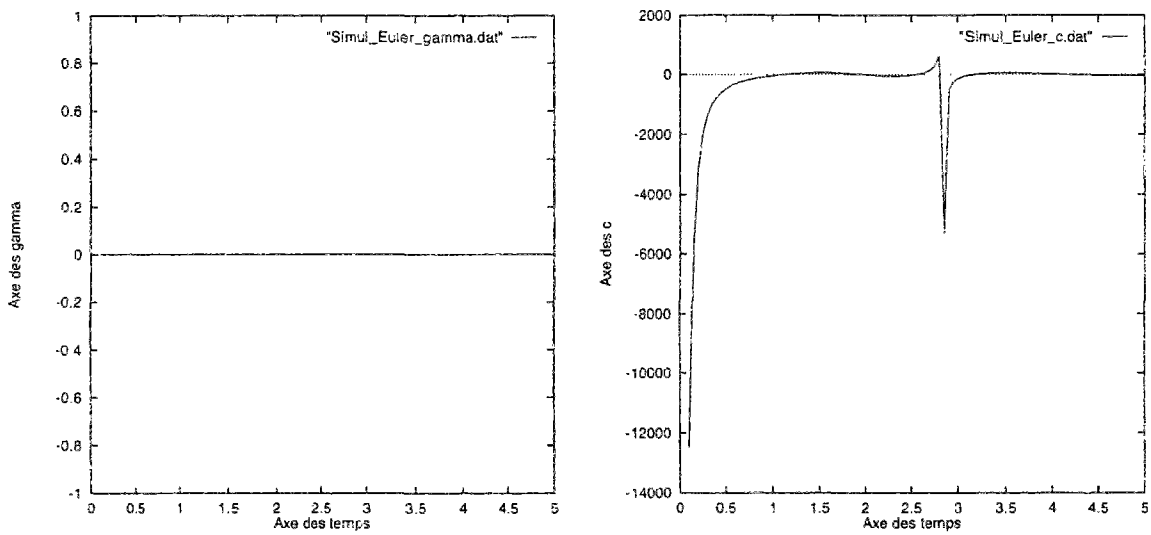


Fig. C.5: *Simulation numérique de la rotation propre duale de l'organe terminal du robot ACMA–H80*

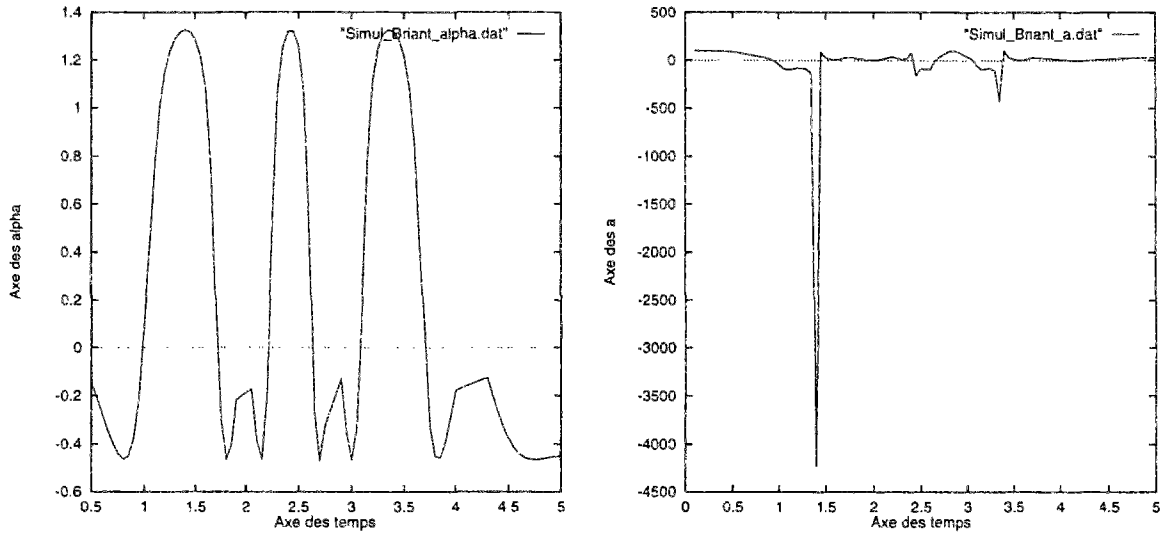


Fig. C.6: *Simulation numérique du roulis dual de l'organe terminal du robot ACMA–H80*

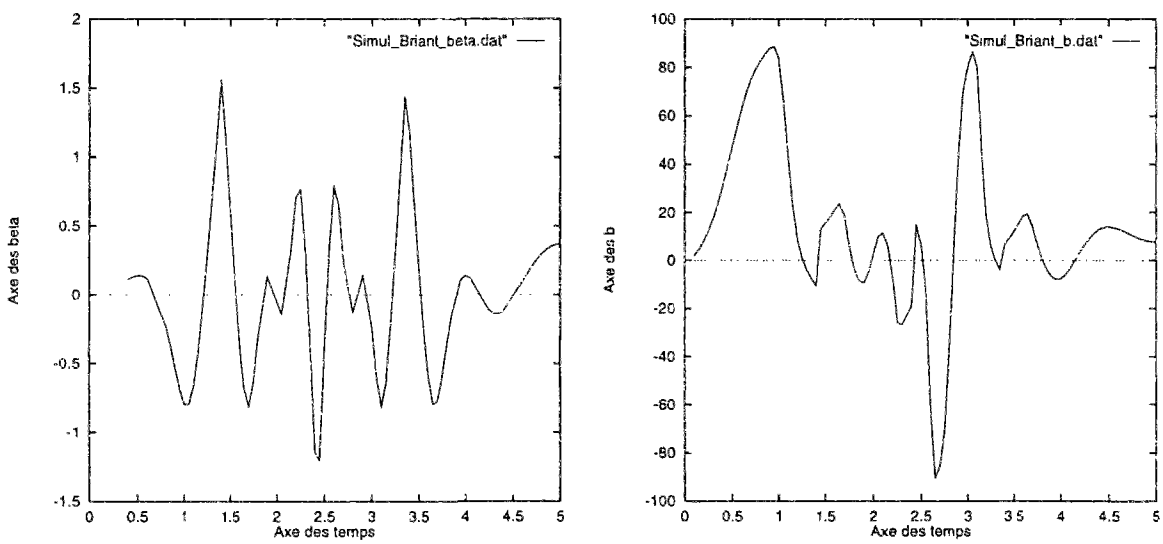


Fig. C.7: *Simulation numérique du tangage dual de l'organe terminal du robot ACMA–H80*

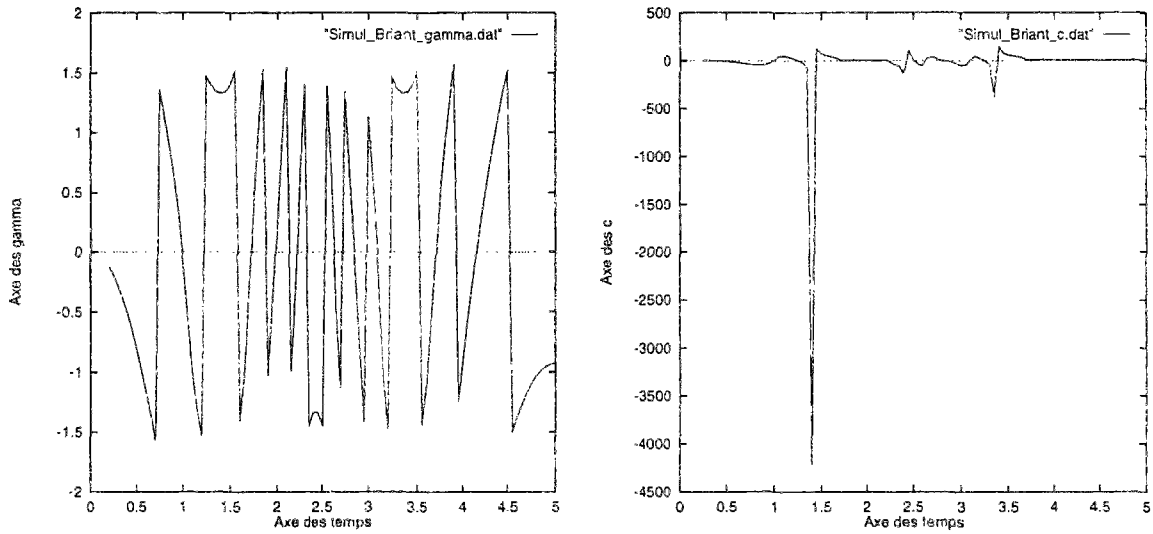


Fig. C.8: *Simulation numérique du lacet duale de l'organe terminal du robot ACMA-H80*

Annexe D

LE ROBOT PUMA 560

D.1 Introduction

Nous présentons dans cet annexe un exemple complet traité par le système *MEDUSA MF77* pour la générations des codes associés au modèle dynamique du robot PUMA 560.

D.2 Caractéristiques

Le Puma 560 est un robot à six degrés de liberté de type pivot. Les caractéristiques du robot, supposées connues dans le cas présent, sont donnés par le tableau suivant:

– Paramètres de D.H. associés –				
i	γ_i	d_i	l_i	θ_i
1	0	0	0	θ_1
2	$-\pi/2$	0	0	θ_2
3	0	D_3	L_3	θ_3
4	$-\pi/2$	D_4	L_4	θ_4
5	$\pi/2$	0	0	θ_5
6	$-\pi/2$	0	0	θ_6

Les autres caractéristiques (masses, moments d'inertie, produits d'inertie, centres d'inertie des éléments) du robot nécessaires à la description paramétrée seront supposées symboliques.

D.3 Description paramétrée grâce au module *DATASEIZE*

Cette section décrit la saisie de la description paramétrée par le module *DATASEIZE* du système *MEDUSA MF77* :

```
ENTRER L'IDENTIFICATEUR COMMUN DES FICHIERS: knowledge == Puma560;

ENTREZ LE NOMBRE N DE D.D.L. DU ROBOT: N == 6;
```

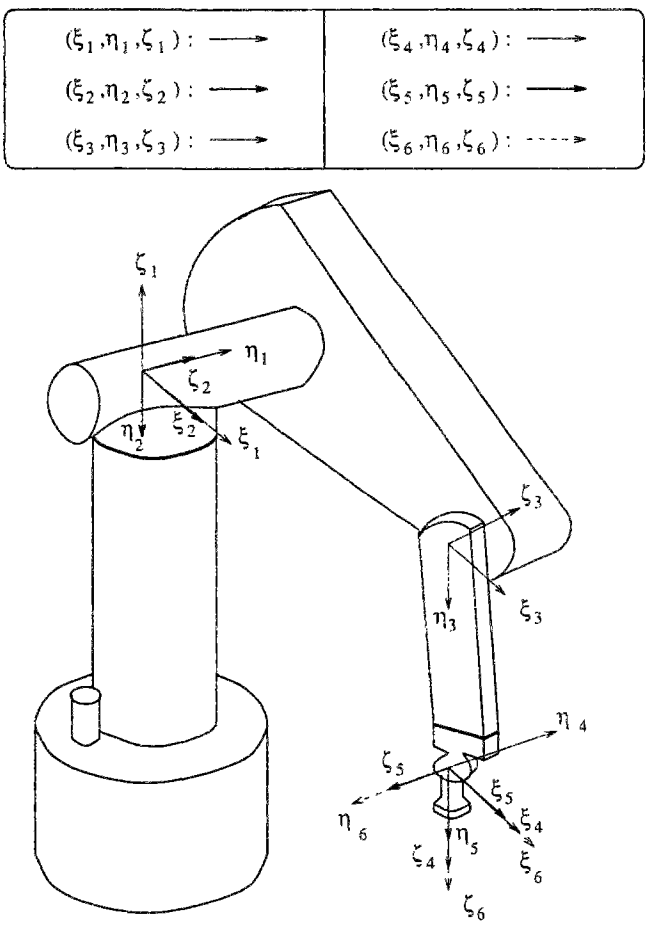


Fig. D.1: Le robot PUMA 560

```
*****************************************
* SAISIE DE LA NATURE DES ARTICULATIONS *
*****************************************
```

```
NATURE DE LA 1e ARTICULATION: JoiNature1 == pivot;

NATURE DE LA 2e ARTICULATION: JoiNature2 == pivot;

NATURE DE LA 3e ARTICULATION: JoiNature3 == pivot;
```

```
NATURE DE LA 4e ARTICULATION: JoiNature4 == pivot;

NATURE DE LA 5e ARTICULATION: JoiNature5 == pivot;

NATURE DE LA 6e ARTICULATION: JoiNature6 == pivot;

                    ****************************************
                    *  SAISIE DES PARAMETRES NON VARIABLES  *
                    *  CARACTERISANT LES ELEMENTS DU ROBOT  *
                    ****************************************

                     ENTREZ LES CARACTERISTIQUES GEOMETRIQUES

                              DE LA CHAINE ROBOTIQUE.


PARAMETRE ANGULAIRE CONSTANT DU 1e CORPS: gamma1 == 0;
--------------------------------------------------------------------------------
1e PARAMETRE EUCLIDIEN CONSTANT DU 1e CORPS: l1 == 0;
--------------------------------------------------------------------------------
2d PARAMETRE EUCLIDIEN CONSTANT DU 1e CORPS: d1 == 0;
--------------------------------------------------------------------------------
PARAMETRE ANGULAIRE CONSTANT DU 2e CORPS: gamma2 == -Pi/2;
--------------------------------------------------------------------------------
1e PARAMETRE EUCLIDIEN CONSTANT DU 2e CORPS: l2 == 0;
--------------------------------------------------------------------------------
2d PARAMETRE EUCLIDIEN CONSTANT DU 2e CORPS: d2 == 0;
--------------------------------------------------------------------------------
PARAMETRE ANGULAIRE CONSTANT DU 3e CORPS: gamma3 == 0;
--------------------------------------------------------------------------------
1e PARAMETRE EUCLIDIEN CONSTANT DU 3e CORPS: l3 == L3;
--------------------------------------------------------------------------------
2d PARAMETRE EUCLIDIEN CONSTANT DU 3e CORPS: d3 == D3;
--------------------------------------------------------------------------------
PARAMETRE ANGULAIRE CONSTANT DU 4e CORPS: gamma4 == -Pi/2;
--------------------------------------------------------------------------------
1e PARAMETRE EUCLIDIEN CONSTANT DU 4e CORPS: l4 == L4;
--------------------------------------------------------------------------------
2d PARAMETRE EUCLIDIEN CONSTANT DU 4e CORPS: d4 == D4;
--------------------------------------------------------------------------------
PARAMETRE ANGULAIRE CONSTANT DU 5e CORPS: gamma5 == Pi/2;
--------------------------------------------------------------------------------
1e PARAMETRE EUCLIDIEN CONSTANT DU 5e CORPS: l5 == 0;
--------------------------------------------------------------------------------
2d PARAMETRE EUCLIDIEN CONSTANT DU 5e CORPS: d5 == 0;
--------------------------------------------------------------------------------
PARAMETRE ANGULAIRE CONSTANT DU 6e CORPS: gamma6 == -Pi/2;
--------------------------------------------------------------------------------
1e PARAMETRE EUCLIDIEN CONSTANT DU 6e CORPS: l6 == 0;
--------------------------------------------------------------------------------
2d PARAMETRE EUCLIDIEN CONSTANT DU 6e CORPS: d6 == 0;
```

```
                    DEBUT DE LA GENERTION DES CODES EN

                              MAPLE ET FORTRAN 77

#  REECRITURE ...
#  REECRITURE ...
#  REECRITURE ...
#  REECRITURE ...
.
.
.
.
#  REECRITURE ...
#  REECRITURE ...
#  REECRITURE ...
#  REECRITURE ...
#
# LA BASE DE CONNAISSANCE CONTENANT LE SYSTEME DE REECRITURE EN CODE MAPLE
# DU MODELE DYNAMIQUE DU ROBOT EST DANS LE FICHIER:   Puma560.map ...
#
# LES SOURCES DU CODE FORTRAN OPTIMISE POUR LE CALCUL DU MODELE DYNAMIQUE
# EST DANS LE FICHIER:   Puma560.for ...
#
#
              VOULEZ VOUS AVOIR LES EQUATIONS EXPLICITES DU MODELE

            DU ROBOT POUR D'EVENTUELLES TRANSFORMATIONS FORMELLES ?

REPONDEZ PAR (o\n): Reponse == n;
```

D.4 Base de connaissance en code maple générée par *MEDUSA MF77*

Cette base de connaissance est sous forme d'un code reconnaissable par le système Maple, formée de règles représentant les références objectives des objets nécessaires à la définition d'un schéma itératif pour le calcul du modèle dynamique du robot Puma 560.

```
#
# -------- Knowledge based on systems of rules rewriting termes --------
#
#
#  The rewrite rule :   R_1
#
STE1 :=   sin(q[1])   ;
#
#  The rewrite rule :   R_2
#
CTE1 :=   cos(q[1])   ;
#
#  The rewrite rule :   R_3
#
STE2 :=   sin(q[2])   ;
```

```
#
#  The rewrite rule :   R_4
#
CTE2 :=   cos(q[2])   ;
#
#  The rewrite rule :   R_5
#
STE3 :=   sin(q[3])   ;
#
#  The rewrite rule :   R_6
#
CTE3 :=   cos(q[3])   ;
#
#  The rewrite rule :   R_7
#
STE4 :=   sin(q[4])   ;
#
#  The rewrite rule :   R_8
#
CTE4 :=   cos(q[4])   ;
#
#  The rewrite rule :   R_9
#
STE5 :=   sin(q[5])   ;
#
#  The rewrite rule :   R_10
#
CTE5 :=   cos(q[5])   ;
#
#  The rewrite rule :   R_11
#
STE6 :=   sin(q[6])   ;
#
#  The rewrite rule :   R_12
#
CTE6 :=   cos(q[6])   ;
#
#  The rewrite rule :   R_13
#
L113 :=   STE3*D3   ;
#
#  The rewrite rule :   R_14
#
L123 :=   -CTE3*D3   ;
#
#  The rewrite rule :   R_15
#
L133 :=   0   ;
#
#  The rewrite rule :   R_16
#
L213 :=   CTE3*D3   ;
```

```
#
#  The rewrite rule :   R_17
#
L223 :=   STE3*D3   ;
#
#  The rewrite rule :   R_18
#
L233 :=   -1.*L3   ;
#
#  The rewrite rule :   R_19
#
L323 :=   L3*CTE3   ;
#
#  The rewrite rule :   R_20
#
L333 :=   0   ;
#
#  The rewrite rule :   R_21
#
L114 :=   STE4*D4   ;
#
#  The rewrite rule :   R_22
#
L124 :=   -CTE4*D4   ;
#
#  The rewrite rule :   R_23
#
L134 :=   0   ;
#
#  The rewrite rule :   R_24
#
L214 :=   L4*STE4   ;
#
#  The rewrite rule :   R_25
#
L224 :=   -L4*CTE4   ;
#
#  The rewrite rule :   R_26
#
L234 :=   0   ;
#
#  The rewrite rule :   R_27
#
L324 :=   STE4*D4   ;
#
#  The rewrite rule :   R_28
#
L334 :=   -1.*L4   ;
#
#  The rewrite rule :   R_29
#
om1 :=   array(1 .. 3,[(1)=0,(2)=0,(3)=Dq[1]])   ;
```

```
#
#  The rewrite rule :   R_30
#
om2 :=   array(1 .. 3,[(1)=-1.*STE2*om1[3],(2)=CTE2*om1[3],(3)=Dq[2]])   ;
#
#  The rewrite rule :   R_31
#
om3 :=   array(1 .. 3,[(1)=CTE3*om2[1]-1.*STE3*om2[2],(2)=STE3*om2[1]+CTE3*om2[
2],(3)=om2[3]+Dq[3]])   ;
#
#  The rewrite rule :   R_32
#
om4 :=   array(1 .. 3,[(1)=CTE4*om3[1]-1.*STE4*om3[3],(2)=STE4*om3[1]+CTE4*om3[
3],(3)=-1.*om3[2]+Dq[4]])   ;
#
#  The rewrite rule :   R_33
#
om5 :=   array(1 .. 3,[(1)=CTE5*om4[1]+STE5*om4[3],(2)=STE5*om4[1]-1.*CTE5*om4[
3],(3)=om4[2]+Dq[5]])   ;
#
#  The rewrite rule :   R_34
#
om6 :=   array(1 .. 3,[(1)=CTE6*om5[1]-1.*STE6*om5[3],(2)=STE6*om5[1]+CTE6*om5[
3],(3)=-1.*om5[2]+Dq[6]])   ;
#
#  The rewrite rule :   R_35
#
vl1:=    array(1 .. 3,[(1)=0,(2)=0,(3)=0])   ;
#
#  The rewrite rule :   R_36
#
vl2:=    array(1 .. 3,[(1)=0,(2)=0,(3)=0])   ;
#
#  The rewrite rule :   R_37
#
vl3:=    array(1 .. 3,[(1)=L113*om2[1]+L213*om2[2]+L313*om2[3],(2)=L123*om2[1]+
L223*om2[2]+L323*om2[3],(3)=L233*om2[2]])   ;
#
#  The rewrite rule :   R_38
#
vl4:=    array(1 .. 3,[(1)=L114*om3[1]+L214*om3[2]+L314*om3[3]+CTE4*vl3[1]-1.*
STE4*vl3[3],(2)=L124*om3[1]+L224*om3[2]+L324*om3[3]+STE4*vl3[1]+CTE4*vl3[3],(3)
=L334*om3[3]-1.*vl3[2]])   ;
#
#  The rewrite rule :   R_39
#
vl5:=    array(1 .. 3,[(1)=CTE5*vl4[1]+STE5*vl4[3],(2)=STE5*vl4[1]-1.*CTE5*vl4[
3],(3)=vl4[2]])   ;
#
#  The rewrite rule :   R_40
#
vl6:=    array(1 .. 3,[(1)=CTE6*vl5[1]-1.*STE6*vl5[3],(2)=STE6*vl5[1]+CTE6*vl5[
```

```
3],(3)=-1.*vl5[2]])   ;
#
#  The rewrite rule :   R_41
#
Dom1 :=   array(1 .. 3,[(1)=0,(2)=0,(3)=DDq[1]])   ;
#
#  The rewrite rule :   R_42
#
Dom2 :=   array(1 .. 3,[(1)=Dq[2]*CTE2*om1[3]-1.*STE2*Dom1[3],(2)=Dq[2]*STE2*
om1[3]+CTE2*Dom1[3],(3)=DDq[2]])   ;
#
#  The rewrite rule :   R_43
#
Dom3 :=   array(1 .. 3,[(1)=Dq[3]*STE3*om2[1]+Dq[3]*CTE3*om2[2]+CTE3*Dom2[1]-1.
*STE3*Dom2[2],(2)=-1.*Dq[3]*CTE3*om2[1]+Dq[3]*STE3*om2[2]+STE3*Dom2[1]+CTE3*
Dom2[2],(3)=Dom2[3]+DDq[3]])   ;
#
#  The rewrite rule :   R_44
#
Dom4 :=   array(1 .. 3,[(1)=Dq[4]*STE4*om3[1]+Dq[4]*CTE4*om3[3]+CTE4*Dom3[1]-1.
*STE4*Dom3[3],(2)=-1.*Dq[4]*CTE4*om3[1]+Dq[4]*STE4*om3[3]+STE4*Dom3[1]+CTE4*
Dom3[3],(3)=-1.*Dom3[2]+DDq[4]])   ;
#
#  The rewrite rule :   R_45
#
Dom5 :=   array(1 .. 3,[(1)=Dq[5]*STE5*om4[1]-1.*Dq[5]*CTE5*om4[3]+CTE5*Dom4[1]
+STE5*Dom4[3],(2)=-1.*Dq[5]*CTE5*om4[1]-1.*Dq[5]*STE5*om4[3]+STE5*Dom4[1]-1.*
CTE5*Dom4[3],(3)=Dom4[2]+DDq[5]])   ;
#
#  The rewrite rule :   R_46
#
Dom6 :=   array(1 .. 3,[(1)=Dq[6]*STE6*om5[1]+Dq[6]*CTE6*om5[3]+CTE6*Dom5[1]-1.
*STE6*Dom5[3],(2)=-1.*Dq[6]*CTE6*om5[1]+Dq[6]*STE6*om5[3]+STE6*Dom5[1]+CTE6*
Dom5[3],(3)=-1.*Dom5[2]+DDq[6]])   ;
#
#  The rewrite rule :   R_47
#
W1 :=   array(1 .. 3,[(1)=0,(2)=0,(3)=gravity])   ;
#
#  The rewrite rule :   R_48
#
W2 :=   array(1 .. 3,[(1)=-1.*STE2*W1[3],(2)=CTE2*W1[3],(3)=0])   ;
#
#  The rewrite rule :   R_49
#
W3 :=   array(1 .. 3,[(1)=CTE3*W2[1]-1.*STE3*W2[2]+L113*Dom2[1]+L213*Dom2[2]+
L313*Dom2[3]+Dq[3]*L123*om2[1]+Dq[3]*L223*om2[2]+Dq[3]*L323*om2[3],(2)=STE3*W2[
1]+CTE3*W2[2]+L123*Dom2[1]+L223*Dom2[2]+L323*Dom2[3]-Dq[3]*L113*om2[1]-Dq[3]*
L213*om2[2]-Dq[3]*L313*om2[3],(3)=L233*Dom2[2]])   ;
#
#  The rewrite rule :   R_50
#
```

```
W4 :=   array(1 .. 3,[(1)=CTE4*W3[1]-1.*STE4*W3[3]+L114*Dom3[1]+L214*Dom3[2]+
L314*Dom3[3]+Dq[4]*L124*om3[1]+Dq[4]*L224*om3[2]+Dq[4]*L324*om3[3]+Dq[4]*STE4*
v13[1]+Dq[4]*CTE4*v13[3],(2)=STE4*W3[1]+CTE4*W3[3]+L124*Dom3[1]+L224*Dom3[2]+
L324*Dom3[3]-1.*Dq[4]*L114*om3[1]-1.*Dq[4]*L214*om3[2]-1.*Dq[4]*L314*om3[3]-1.*
Dq[4]*CTE4*v13[1]+Dq[4]*STE4*v13[3],(3)=-1.*W3[2]+L334*Dom3[3]])   ;
#
#  The rewrite rule :   R_51
#
W5 :=   array(1 .. 3,[(1)=CTE5*W4[1]+STE5*W4[3]+Dq[5]*STE5*v14[1]-1.*Dq[5]*CTE5
*v14[3],(2)=STE5*W4[1]-1.*CTE5*W4[3]-1.*Dq[5]*CTE5*v14[1]-1.*Dq[5]*STE5*v14[3],
(3)=W4[2]])   ;
#
#  The rewrite rule :   R_52
#
W6 :=   array(1 .. 3,[(1)=CTE6*W5[1]-1.*STE6*W5[3]+Dq[6]*STE6*v15[1]+Dq[6]*CTE6
*v15[3],(2)=STE6*W5[1]+CTE6*W5[3]-1.*Dq[6]*CTE6*v15[1]+Dq[6]*STE6*v15[3],(3)=-1
.*W5[2]])   ;
#
#  The rewrite rule :   R_53
#
VOG1 :=   array(1 .. 3,[(1)=-om1[3]*ros1[2],(2)=om1[3]*ros1[1],(3)=0])   ;
#
#  The rewrite rule :   R_54
#
Fe1 :=   array(1 .. 3,[(1)=-m1*(Dom1[3]*ros1[2]+om1[3]*VOG1[2]),(2)=m1*(Dom1[3]
*ros1[1]+om1[3]*VOG1[1]),(3)=m1*W1[3]])   ;
#
#  The rewrite rule :   R_55
#
VOG2 :=   array(1 .. 3,[(1)=om2[2]*ros2[3]-om2[3]*ros2[2],(2)=om2[3]*ros2[1]-
om2[1]*ros2[3],(3)=om2[1]*ros2[2]-om2[2]*ros2[1]])   ;
#
#  The rewrite rule :   R_56
#
Fe2 :=   array(1 .. 3,[(1)=-m2*(-W2[1]-Dom2[2]*ros2[3]+Dom2[3]*ros2[2]-om2[2]*
VOG2[3]+om2[3]*VOG2[2]),(2)=m2*(W2[2]+Dom2[3]*ros2[1]-Dom2[1]*ros2[3]+om2[3]*
VOG2[1]-om2[1]*VOG2[3]),(3)=m2*(Dom2[1]*ros2[2]-Dom2[2]*ros2[1]+om2[1]*VOG2[2]-
om2[2]*VOG2[1])])   ;
#
#  The rewrite rule :   R_57
#
VOG3 :=   array(1 .. 3,[(1)=v13[1]+om3[2]*ros3[3]-om3[3]*ros3[2],(2)=v13[2]+om3
[3]*ros3[1]-om3[1]*ros3[3],(3)=v13[3]+om3[1]*ros3[2]-om3[2]*ros3[1]])   ;
#
#  The rewrite rule :   R_58
#
Fe3 :=   array(1 .. 3,[(1)=-m3*(-W3[1]-Dom3[2]*ros3[3]+Dom3[3]*ros3[2]-om3[2]*
VOG3[3]+om3[3]*VOG3[2]),(2)=m3*(W3[2]+Dom3[3]*ros3[1]-Dom3[1]*ros3[3]+om3[3]*
VOG3[1]-om3[1]*VOG3[3]),(3)=-m3*(-W3[3]-Dom3[1]*ros3[2]+Dom3[2]*ros3[1]-om3[1]*
VOG3[2]+om3[2]*VOG3[1])])   ;
#
#  The rewrite rule :   R_59
```

```
#
VOG4 :=   array(1 .. 3,[(1)=vl4[1]+om4[2]*ros4[3]-om4[3]*ros4[2],(2)=vl4[2]+om4
[3]*ros4[1]-om4[1]*ros4[3],(3)=vl4[3]+om4[1]*ros4[2]-om4[2]*ros4[1]])   ;
#
#  The rewrite rule :   R_60
#
Fe4 :=   array(1 .. 3,[(1)=m4*(W4[1]+Dom4[2]*ros4[3]-Dom4[3]*ros4[2]+om4[2]*
VOG4[3]-om4[3]*VOG4[2]),(2)=m4*(W4[2]+Dom4[3]*ros4[1]-Dom4[1]*ros4[3]+om4[3]*
VOG4[1]-om4[1]*VOG4[3]),(3)=m4*(W4[3]+Dom4[1]*ros4[2]-Dom4[2]*ros4[1]+om4[1]*
VOG4[2]-om4[2]*VOG4[1])])   ;
#
#  The rewrite rule :   R_61
#
VOG5 :=   array(1 .. 3,[(1)=vl5[1]+om5[2]*ros5[3]-om5[3]*ros5[2],(2)=vl5[2]+om5
[3]*ros5[1]-om5[1]*ros5[3],(3)=vl5[3]+om5[1]*ros5[2]-om5[2]*ros5[1]])   ;
#
#  The rewrite rule :   R_62
#
Fe5 :=   array(1 .. 3,[(1)=m5*(W5[1]+Dom5[2]*ros5[3]-Dom5[3]*ros5[2]+om5[2]*
VOG5[3]-om5[3]*VOG5[2]),(2)=-m5*(-W5[2]-Dom5[3]*ros5[1]+Dom5[1]*ros5[3]-om5[3]*
VOG5[1]+om5[1]*VOG5[3]),(3)=m5*(W5[3]+Dom5[1]*ros5[2]-Dom5[2]*ros5[1]+om5[1]*
VOG5[2]-om5[2]*VOG5[1])])   ;
#
#  The rewrite rule :   R_63
#
VOG6 :=   array(1 .. 3,[(1)=vl6[1]+om6[2]*ros6[3]-om6[3]*ros6[2],(2)=vl6[2]+om6
[3]*ros6[1]-om6[1]*ros6[3],(3)=vl6[3]+om6[1]*ros6[2]-om6[2]*ros6[1]])   ;
#
#  The rewrite rule :   R_64
#
Fe6 :=   array(1 .. 3,[(1)=m6*(W6[1]+Dom6[2]*ros6[3]-Dom6[3]*ros6[2]+om6[2]*
VOG6[3]-om6[3]*VOG6[2]),(2)=m6*(W6[2]+Dom6[3]*ros6[1]-Dom6[1]*ros6[3]+om6[3]*
VOG6[1]-om6[1]*VOG6[3]),(3)=-m6*(-W6[3]-Dom6[1]*ros6[2]+Dom6[2]*ros6[1]-om6[1]*
VOG6[2]+om6[2]*VOG6[1])])   ;
#
#  The rewrite rule :   R_65
#
IOM1 :=   array(1 .. 3,[(1)=-Ixz[1]*om1[3],(2)=-Iyz[1]*om1[3],(3)=Izz[1]*om1[3]
])   ;
#
#  The rewrite rule :   R_66
#
Ne1 :=   array(1 .. 3,[(1)=-Ixz[1]*Dom1[3]-om1[3]*IOM1[2]+ros1[2]*Fe1[3]-ros1[3
]*Fe1[2],(2)=-Iyz[1]*Dom1[3]+om1[3]*IOM1[1]+ros1[3]*Fe1[1]-ros1[1]*Fe1[3],(3)=
Izz[1]*Dom1[3]+ros1[1]*Fe1[2]-ros1[2]*Fe1[1]])   ;
#
#  The rewrite rule :   R_67
#
IOM2 :=   array(1 .. 3,[(1)=Ixx[2]*om2[1]-Ixy[2]*om2[2]-Ixz[2]*om2[3],(2)=-Ixy[
2]*om2[1]+Iyy[2]*om2[2]-Iyz[2]*om2[3],(3)=-Ixz[2]*om2[1]-Iyz[2]*om2[2]+Izz[2]*
om2[3]])   ;
#
```

```
#  The rewrite rule :   R_68
#
Ne2 :=   array(1 .. 3,[(1)=Ixx[2]*Dom2[1]-Ixy[2]*Dom2[2]-Ixz[2]*Dom2[3]+om2[2]*
IOM2[3]-om2[3]*IOM2[2]+ros2[2]*Fe2[3]-ros2[3]*Fe2[2],(2)=-Ixy[2]*Dom2[1]+Iyy[2]
*Dom2[2]-Iyz[2]*Dom2[3]+om2[3]*IOM2[1]-om2[1]*IOM2[3]+ros2[3]*Fe2[1]-ros2[1]*
Fe2[3],(3)=-Ixz[2]*Dom2[1]-Iyz[2]*Dom2[2]+Izz[2]*Dom2[3]+om2[1]*IOM2[2]-om2[2]*
IOM2[1]+ros2[1]*Fe2[2]-ros2[2]*Fe2[1]])   ;
#
#  The rewrite rule :   R_69
#
IOM3 :=   array(1 .. 3,[(1)=Ixx[3]*om3[1]-Ixy[3]*om3[2]-Ixz[3]*om3[3],(2)=-Ixy[
3]*om3[1]+Iyy[3]*om3[2]-Iyz[3]*om3[3],(3)=-Ixz[3]*om3[1]-Iyz[3]*om3[2]+Izz[3]*
om3[3]])   ;
#
#  The rewrite rule :   R_70
#
Ne3 :=   array(1 .. 3,[(1)=Ixx[3]*Dom3[1]-Ixy[3]*Dom3[2]-Ixz[3]*Dom3[3]+om3[2]*
IOM3[3]-om3[3]*IOM3[2]+ros3[2]*Fe3[3]-ros3[3]*Fe3[2],(2)=-Ixy[3]*Dom3[1]+Iyy[3]
*Dom3[2]-Iyz[3]*Dom3[3]+om3[3]*IOM3[1]-om3[1]*IOM3[3]+ros3[3]*Fe3[1]-ros3[1]*
Fe3[3],(3)=-Ixz[3]*Dom3[1]-Iyz[3]*Dom3[2]+Izz[3]*Dom3[3]+om3[1]*IOM3[2]-om3[2]*
IOM3[1]+ros3[1]*Fe3[2]-ros3[2]*Fe3[1]])   ;
#
#  The rewrite rule :   R_71
#
IOM4 :=   array(1 .. 3,[(1)=Ixx[4]*om4[1]-Ixy[4]*om4[2]-Ixz[4]*om4[3],(2)=-Ixy[
4]*om4[1]+Iyy[4]*om4[2]-Iyz[4]*om4[3],(3)=-Ixz[4]*om4[1]-Iyz[4]*om4[2]+Izz[4]*
om4[3]])   ;
#
#  The rewrite rule :   R_72
#
Ne4 :=   array(1 .. 3,[(1)=Ixx[4]*Dom4[1]-Ixy[4]*Dom4[2]-Ixz[4]*Dom4[3]+om4[2]*
IOM4[3]-om4[3]*IOM4[2]+ros4[2]*Fe4[3]-ros4[3]*Fe4[2],(2)=-Ixy[4]*Dom4[1]+Iyy[4]
*Dom4[2]-Iyz[4]*Dom4[3]+om4[3]*IOM4[1]-om4[1]*IOM4[3]+ros4[3]*Fe4[1]-ros4[1]*
Fe4[3],(3)=-Ixz[4]*Dom4[1]-Iyz[4]*Dom4[2]+Izz[4]*Dom4[3]+om4[1]*IOM4[2]-om4[2]*
IOM4[1]+ros4[1]*Fe4[2]-ros4[2]*Fe4[1]])   ;
#
#  The rewrite rule :   R_73
#
IOM5 :=   array(1 .. 3,[(1)=Ixx[5]*om5[1]-Ixy[5]*om5[2]-Ixz[5]*om5[3],(2)=-Ixy[
5]*om5[1]+Iyy[5]*om5[2]-Iyz[5]*om5[3],(3)=-Ixz[5]*om5[1]-Iyz[5]*om5[2]+Izz[5]*
om5[3]])   ;
#
#  The rewrite rule :   R_74
#
Ne5 :=   array(1 .. 3,[(1)=Ixx[5]*Dom5[1]-Ixy[5]*Dom5[2]-Ixz[5]*Dom5[3]+om5[2]*
IOM5[3]-om5[3]*IOM5[2]+ros5[2]*Fe5[3]-ros5[3]*Fe5[2],(2)=-Ixy[5]*Dom5[1]+Iyy[5]
*Dom5[2]-Iyz[5]*Dom5[3]+om5[3]*IOM5[1]-om5[1]*IOM5[3]+ros5[3]*Fe5[1]-ros5[1]*
Fe5[3],(3)=-Ixz[5]*Dom5[1]-Iyz[5]*Dom5[2]+Izz[5]*Dom5[3]+om5[1]*IOM5[2]-om5[2]*
IOM5[1]+ros5[1]*Fe5[2]-ros5[2]*Fe5[1]])   ;
#
#  The rewrite rule :   R_75
#
```

```
IOM6 :=   array(1 .. 3,[(1)=Ixx[6]*om6[1]-Ixy[6]*om6[2]-Ixz[6]*om6[3],(2)=-Ixy[
6]*om6[1]+Iyy[6]*om6[2]-Iyz[6]*om6[3],(3)=-Ixz[6]*om6[1]-Iyz[6]*om6[2]+Izz[6]*
om6[3]])   ;
#
#  The rewrite rule :   R_76
#
Ne6 :=   array(1 .. 3,[(1)=Ixx[6]*Dom6[1]-Ixy[6]*Dom6[2]-Ixz[6]*Dom6[3]+om6[2]*
IOM6[3]-om6[3]*IOM6[2]+ros6[2]*Fe6[3]-ros6[3]*Fe6[2],(2)=-Ixy[6]*Dom6[1]+Iyy[6]
*Dom6[2]-Iyz[6]*Dom6[3]+om6[3]*IOM6[1]-om6[1]*IOM6[3]+ros6[3]*Fe6[1]-ros6[1]*
Fe6[3],(3)=-Ixz[6]*Dom6[1]-Iyz[6]*Dom6[2]+Izz[6]*Dom6[3]+om6[1]*IOM6[2]-om6[2]*
IOM6[1]+ros6[1]*Fe6[2]-ros6[2]*Fe6[1]])   ;
#
#  The rewrite rule :   R_77
#
fa6 :=   array(1 .. 3,[(1)=Fe6[1],(2)=Fe6[2],(3)=Fe6[3]])   ;
#
#  The rewrite rule :   R_78
#
fa5 :=   array(1 .. 3,[(1)=CTE6*fa6[1]+STE6*fa6[2]+Fe5[1],(2)=-1.*fa6[3]+Fe5[2]
,(3)=-1.*STE6*fa6[1]+CTE6*fa6[2]+Fe5[3]])   ;
#
#  The rewrite rule :   R_79
#
fa4 :=   array(1 .. 3,[(1)=CTE5*fa5[1]+STE5*fa5[2]+Fe4[1],(2)=fa5[3]+Fe4[2],(3)
=STE5*fa5[1]-1.*CTE5*fa5[2]+Fe4[3]])   ;
#
#  The rewrite rule :   R_80
#
fa3 :=   array(1 .. 3,[(1)=CTE4*fa4[1]+STE4*fa4[2]+Fe3[1],(2)=-1.*fa4[3]+Fe3[2]
,(3)=-1.*STE4*fa4[1]+CTE4*fa4[2]+Fe3[3]])   ;
#
#  The rewrite rule :   R_81
#
fa2 :=   array(1 .. 3,[(1)=CTE3*fa3[1]+STE3*fa3[2]+Fe2[1],(2)=-1.*STE3*fa3[1]+
CTE3*fa3[2]+Fe2[2],(3)=fa3[3]+Fe2[3]])   ;
#
#  The rewrite rule :   R_82
#
fa1 :=   array(1 .. 3,[(1)=CTE2*fa2[1]+STE2*fa2[2]+Fe1[1],(2)=-1.*fa2[3]+Fe1[2]
,(3)=-1.*STE2*fa2[1]+CTE2*fa2[2]+Fe1[3]])   ;
#
#  The rewrite rule :   R_83
#
na6 :=   array(1 .. 3,[(1)=Ne6[1],(2)=Ne6[2],(3)=Ne6[3]])   ;
#
#  The rewrite rule :   R_84
#
NFR5 :=   array(1 .. 3,[(1)=na6[1],(2)=na6[2],(3)=na6[3]])   ;
#
#  The rewrite rule :   R_85
#
na5 :=   array(1 .. 3,[(1)=Ne5[1]+CTE6*NFR5[1]+STE6*NFR5[2],(2)=Ne5[2]-1.*NFR5[
```

```
3],(3)=Ne5[3]-1.*STE6*NFR5[1]+CTE6*NFR5[2]])   ;
#
#  The rewrite rule :   R_86
#
NFR4 :=   array(1 .. 3,[(1)=na5[1],(2)=na5[2],(3)=na5[3]])   ;
#
#  The rewrite rule :   R_87
#
na4 :=   array(1 .. 3,[(1)=Ne4[1]+CTE5*NFR4[1]+STE5*NFR4[2],(2)=Ne4[2]+NFR4[3],
(3)=Ne4[3]+STE5*NFR4[1]-1.*CTE5*NFR4[2]])   ;
#
#  The rewrite rule :   R_88
#
NFR3 :=   array(1 .. 3,[(1)=na4[1]+L114*fa4[1]+L214*fa4[2]+L314*fa4[3],(2)=na4[
2]+L124*fa4[1]+L224*fa4[2]+L324*fa4[3],(3)=na4[3]+L334*fa4[3]])   ;
#
#  The rewrite rule :   R_89
#
na3 :=   array(1 .. 3,[(1)=Ne3[1]+CTE4*NFR3[1]+STE4*NFR3[2],(2)=Ne3[2]-1.*NFR3[
3],(3)=Ne3[3]-1.*STE4*NFR3[1]+CTE4*NFR3[2]])   ;
#
#  The rewrite rule :   R_90
#
NFR2 :=   array(1 .. 3,[(1)=na3[1]+L113*fa3[1]+L213*fa3[2]+L313*fa3[3],(2)=na3[
2]+L123*fa3[1]+L223*fa3[2]+L323*fa3[3],(3)=na3[3]+L233*fa3[2]])   ;
#
#  The rewrite rule :   R_91
#
na2 :=   array(1 .. 3,[(1)=Ne2[1]+CTE3*NFR2[1]+STE3*NFR2[2],(2)=Ne2[2]-1.*STE3*
NFR2[1]+CTE3*NFR2[2],(3)=Ne2[3]+NFR2[3]])   ;
#
#  The rewrite rule :   R_92
#
NFR1 :=   array(1 .. 3,[(1)=na2[1],(2)=na2[2],(3)=na2[3]])   ;
#
#  The rewrite rule :   R_93
#
na1 :=   array(1 .. 3,[(1)=Ne1[1]+CTE2*NFR1[1]+STE2*NFR1[2],(2)=Ne1[2]-1.*NFR1[
3],(3)=Ne1[3]-1.*STE2*NFR1[1]+CTE2*NFR1[2]])   ;
#
#  The rewrite rule :   R_94
#
Qg1 :=   na1[3]   ;
#
#  The rewrite rule :   R_95
#
Qg2 :=   na2[3]   ;
#
#  The rewrite rule :   R_96
#
Qg3 :=   na3[3]   ;
#
```

```
#  The rewrite rule :   R_97
#
Qg4 :=   na4[3]   ;
#
#  The rewrite rule :   R_98
#
Qg5 :=   na5[3]   ;
#
#  The rewrite rule :   R_99
#
Qg6 :=   na6[3]   ;
```

D.5 Code Fortran généré par *MEDUSA MF77*

Dans cette section, nous présentons le code en ligne dans le langage Fortran généré automatiquement par le système *MEDUSA MF77*. Il présente un schéma de calcul, du modèle dynamique itératif, optimisé en nombre d'opérations élémentaires d'addition et de multiplication et en nombre d'évaluations des fonctions trigonométriques *cos* et *sin*.

```
c
c   ================================================================
c     -- The Fortran file Puma560.f automaticlly generated by
c       MEDUSA MF77 (version 1.0).
c     -- Ali HAMLILI, CERMA-ENPC, Paris, France.
c       E-mail: ha@thalie.enpc.fr,
c       Tel: (1) 43 04 40 93,
c       Fax: (1) 43 04 63 64.
c
c   ================================================================
c
      subroutine Puma560( q, Dq, DDq, Qg)
c
c
       implicite  double precision (a-z)
c
c     -- This subroutine computes the generalized forces and torques
c     needed at the joints in order to balance the reaction forces
c     and moments acting on the links.
c
c     -- The effect of gravity loading on the links is included simply
c     by setting the linear fundamental acceleration W= Gravity,
c     where "Gravity" is the gravity vector.
c
c
c
c     Auxiliary variables
c     -------------------
c
c     Type of variables
        double precision  NFR4(3), IOM2(3), NFR2(3), VOG2(3), IOM3(3), NF
     $  R3(3), VOG3(3), VOG4(3), IOM4(3), CTE1, STE1, CTE2, STE2, CTE3,
```

```
     $  STE3, CTE4, STE4, CTE5, STE5, IOM5(3), NFR5(3), VOG5(3), IOM6(3)
     $  , NFR6(3), VOG6(3), CTE6, STE6
c    Common declarations
c
      common/Vaux/ IOM2, CTE1, STE1, CTE2, STE2, NFR2, VOG2, IOM3, CTE3,
     $   STE3, NFR3, VOG3, IOM4, CTE4, STE4, NFR4, VOG4, IOM5, CTE5, STE
     $  5, NFR5, VOG5, IOM6, CTE6, STE6, NFR6, VOG6
c
c
c    Dynamic model variables
c    -----------------------
c
c    Type of variables
       double precision  q(6), Dq(6), DDq(6), Qg(6), Ixx(6), Iyy(6), Izz
     $  (6), Ixy(6), Ixz(6), Iyz(6), gravity
       double precision  om1(3), Dom1(3), vl1(3), W1(3), Fe1(3), Ne1(3),
     $  fa1(3), na1(3), ros1(3), m1
       double precision  om2(3), Dom2(3), vl2(3), W2(3), Fe2(3), Ne2(3),
     $  fa2(3), na2(3), ros2(3), m2
       double precision  om3(3), Dom3(3), vl3(3), W3(3), Fe3(3), Ne3(3),
     $  fa3(3), na3(3), ros3(3), m3
       double precision  om4(3), Dom4(3), vl4(3), W4(3), Fe4(3), Ne4(3),
     $  fa4(3), na4(3), ros4(3), m4
       double precision  om5(3), Dom5(3), vl5(3), W5(3), Fe5(3), Ne5(3),
     $  fa5(3), na5(3), ros5(3), m5
       double precision  om6(3), Dom6(3), vl6(3), W6(3), Fe6(3), Ne6(3),
     $  fa6(3), na6(3), ros6(3), m6
       double precision  L3,D3,L4,D4
c
c
c
c
c    Common declarations
c
      common/Vgen/ q, Dq, DDq, Qg
c
c
      common/Vcmg/ ros1, m1, Ixy, m4, m5, Iyy, Ixx, m6, ros2, ros3, Izz,
     $   gravity, m3, Ixz, Iyz, ros4, ros5, ros6, m2
c
c
      common/Vmec/ om1, Dom1, vl1, W1, Fe1, Ne1, fa1, na1
     $  , om2, Dom2, vl2, W2, Fe2, Ne2, fa2, na2
     $  , om3, Dom3, vl3, W3, Fe3, Ne3, fa3, na3
     $  , om4, Dom4, vl4, W4, Fe4, Ne4, fa4, na4
     $  , om5, Dom5, vl5, W5, Fe5, Ne5, fa5, na5
     $  , om6, Dom6, vl6, W6, Fe6, Ne6, fa6, na6
c
c
      common/Vscal/L3,D3,L4,D4
c
c
```

```
c
c     -- Where q: the 6 dimensional vector of generalised positions,
c              Dq: the 6 dimensional vector of generalised velocities,
c             DDq: the 6 dimensional vector of generalised accelerations,
c              Qg: the 6 dimensional vector of generalised forces and torques,
c           ros_i: the three dimensional vector position of the center of mass
c                  G_i relative to the origine O_i of frame i attached to
c                  the i_th link,
c            om_i: the angular velocity of link i,
c            vl_i: the linear velocity of link i in the origine O_i,
c           Dom_i: the angular acceleration of link i,
c           W_i: the linear acceleration of link i in
c                  the origine O_i augmented by m_i*Gravity_i,
c                  where m_i is the mass of the i_th body and
c                  Gravity_i is the Gravity vector expressed in the i_th frame,
c            Fe_i: the total external force exrted on link i,
c            Ne_i: the total external moment exerted on link i in
c                  the origine O_i,
c            fa_i: the force exrted on link i by link i-1,
c            na_i: the moment exerted on link i by link i-1 evaluated
c                  in the origine O_i,
c     Ixx,Iyy,Izz: the 6 dimensional vectors of the link mass
c                  menents about the centers of mass,
c     Ixy,Ixz,Iyz: the 6 dimensional vectors of the link mass
c                  products about the origins of frames attached to links,
c              g: the local gravitational parameter.
c
c
          STE1 = dsin(q(1))
          CTE1 = dcos(q(1))
          STE2 = dsin(q(2))
          CTE2 = dcos(q(2))
          STE3 = dsin(q(3))
          CTE3 = dcos(q(3))
          STE4 = dsin(q(4))
          CTE4 = dcos(q(4))
          STE5 = dsin(q(5))
          CTE5 = dcos(q(5))
          STE6 = dsin(q(6))
          CTE6 = dcos(q(6))
          L113 = STE3*D3
          L123 = -CTE3*D3
          L213 = -L123
          L223 = L113
          L233 = -L3
          L313 = -L3*STE3
          L323 = L3*CTE3
c       L333 is equal to zero
c
          L114 = STE4*D4
          L124 = -CTE4*D4
          L214 = L4*STE4
```

```
        L224 = -L4*CTE4
c     L234 is equal to zero
c
        L314 = CTE4*D4
        L324 = STE4*D4
        L334 = -L4
c
c     -- Now the program computes angular velocities.
c
c
c                --------- The 1st link ---------
c
c        *  1st coordinate
c     om1(1) is equal to zero
c
c        *  2nd coordinate
c     om1(2) is equal to zero
c
c        *  3rd coordinate
        om1(3) = Dq(1)
c
c                --------- The 2nd link ---------
c
c        *  1st coordinate
        om2(1) = -STE2*om1(3)
c        *  2nd coordinate
        om2(2) = CTE2*om1(3)
c        *  3rd coordinate
        om2(3) = Dq(2)
c
c                --------- The 3rd link ---------
c
c        *  1st coordinate
        om3(1) = CTE3*om2(1)-STE3*om2(2)
c        *  2nd coordinate
        om3(2) = STE3*om2(1)+CTE3*om2(2)
c        *  3rd coordinate
        om3(3) = om2(3)+Dq(3)
c
c                --------- The 4th link ---------
c
c        *  1st coordinate
        om4(1) = CTE4*om3(1)-STE4*om3(3)
c        *  2nd coordinate
        om4(2) = STE4*om3(1)+CTE4*om3(3)
c        *  3rd coordinate
        om4(3) = -om3(2)+Dq(4)
c
c                --------- The 5th link ---------
c
c        *  1st coordinate
        om5(1) = CTE5*om4(1)+STE5*om4(3)
```

```
c         * 2nd coordinate
          om5(2) = STE5*om4(1)-CTE5*om4(3)
c         * 3rd coordinate
          om5(3) = om4(2)+Dq(5)
c
c                 --------- The 6th link ---------
c
c         * 1st coordinate
          om6(1) = CTE6*om5(1)-STE6*om5(3)
c         * 2nd coordinate
          om6(2) = STE6*om5(1)+CTE6*om5(3)
c         * 3rd coordinate
          om6(3) = -om5(2)+Dq(6)
c
c     -- Now the program computes linear velocities.
c
c
c                 --------- The 1st link ---------
c
c         * 1st coordinate
c    vl1(1) is equal to zero
c
c         * 2nd coordinate
c    vl1(2) is equal to zero
c
c         * 3rd coordinate
c    vl1(3) is equal to zero
c
c
c                 --------- The 2nd link ---------
c
c         * 1st coordinate
c    vl2(1) is equal to zero
c
c         * 2nd coordinate
c    vl2(2) is equal to zero
c
c         * 3rd coordinate
c    vl2(3) is equal to zero
c
c
c                 --------- The 3rd link ---------
c
c         * 1st coordinate
        vl3(1) = L113*om2(1)+L213*om2(2)+L313*om2(3)
c         * 2nd coordinate
        vl3(2) = L123*om2(1)+L223*om2(2)+L323*om2(3)
c         * 3rd coordinate
        vl3(3) = L233*om2(2)
c
c                 --------- The 4th link ---------
c
```

```
c        *  1st coordinate
      vl4(1) = L114*om3(1)+L214*om3(2)+L314*om3(3)+CTE4*vl3(1)-STE4*vl3(
     $ 3)
c        *  2nd coordinate
      vl4(2) = L124*om3(1)+L224*om3(2)+L324*om3(3)+STE4*vl3(1)+CTE4*vl3(
     $ 3)
c        *  3rd coordinate
      vl4(3) = L334*om3(3)-vl3(2)
c
c                --------- The 5th link ---------
c
c        *  1st coordinate
      vl5(1) = CTE5*vl4(1)+STE5*vl4(3)
c        *  2nd coordinate
      vl5(2) = STE5*vl4(1)-CTE5*vl4(3)
c        *  3rd coordinate
      vl5(3) = vl4(2)
c
c                --------- The 6th link ---------
c
c        *  1st coordinate
      vl6(1) = CTE6*vl5(1)-STE6*vl5(3)
c        *  2nd coordinate
      vl6(2) = STE6*vl5(1)+CTE6*vl5(3)
c        *  3rd coordinate
      vl6(3) = -vl5(2)
c
c     -- Now the program computes angular accelerations.
c
c
c                --------- The 1st link ---------
c
c        *  1st coordinate
c     Dom1(1) is equal to zero
c
c        *  2nd coordinate
c     Dom1(2) is equal to zero
c
c        *  3rd coordinate
        Dom1(3) = DDq(1)
c
c                --------- The 2nd link ---------
c
c        *  1st coordinate
        Dom2(1) = Dq(2)*CTE2*om1(3)-STE2*Dom1(3)
c        *  2nd coordinate
        Dom2(2) = Dq(2)*STE2*om1(3)+CTE2*Dom1(3)
c        *  3rd coordinate
        Dom2(3) = DDq(2)
c
c                --------- The 3rd link ---------
c
```

```
c         *  1st coordinate
          Dom3(1) = Dq(3)*STE3*om2(1)+Dq(3)*CTE3*om2(2)+CTE3*Dom2(1)-STE3*
     $ Dom2(2)
c         *  2nd coordinate
          Dom3(2) = -Dq(3)*CTE3*om2(1)+Dq(3)*STE3*om2(2)+STE3*Dom2(1)+CTE3
     $ *Dom2(2)
c         *  3rd coordinate
          Dom3(3) = Dom2(3)+DDq(3)
c
c                 --------- The 4th link ---------
c
c         *  1st coordinate
          Dom4(1) = Dq(4)*STE4*om3(1)+Dq(4)*CTE4*om3(3)+CTE4*Dom3(1)-STE4*
     $ Dom3(3)
c         *  2nd coordinate
          Dom4(2) = -Dq(4)*CTE4*om3(1)+Dq(4)*STE4*om3(3)+STE4*Dom3(1)+CTE4
     $ *Dom3(3)
c         *  3rd coordinate
          Dom4(3) = -Dom3(2)+DDq(4)
c
c                 --------- The 5th link ---------
c
c         *  1st coordinate
          Dom5(1) = Dq(5)*STE5*om4(1)-Dq(5)*CTE5*om4(3)+CTE5*Dom4(1)+STE5*
     $ Dom4(3)
c         *  2nd coordinate
          Dom5(2) = -Dq(5)*CTE5*om4(1)-Dq(5)*STE5*om4(3)+STE5*Dom4(1)-CTE5
     $ *Dom4(3)
c         *  3rd coordinate
          Dom5(3) = Dom4(2)+DDq(5)
c
c                 --------- The 6th link ---------
c
c         *  1st coordinate
          Dom6(1) = Dq(6)*STE6*om5(1)+Dq(6)*CTE6*om5(3)+CTE6*Dom5(1)-STE6*
     $ Dom5(3)
c         *  2nd coordinate
          Dom6(2) = -Dq(6)*CTE6*om5(1)+Dq(6)*STE6*om5(3)+STE6*Dom5(1)+CTE6
     $ *Dom5(3)
c         *  3rd coordinate
          Dom6(3) = -Dom5(2)+DDq(6)
c
c     -- Now the program computes linear fundamental accelerations.
c
c
c                 --------- The 1st link ---------
c
c         *  1st coordinate
c      W1(1) is equal to zero
c
c         *  2nd coordinate
c      W1(2) is equal to zero
```

```
c
c        *  3rd coordinate
         W1(3) = gravity
c
c                 --------- The 2nd link ---------
c
c        *  1st coordinate
         W2(1) = -STE2*W1(3)
c        *  2nd coordinate
         W2(2) = CTE2*W1(3)
c        *  3rd coordinate
c     W2(3) is equal to zero
c
c
c                 --------- The 3rd link ---------
c
c        *  1st coordinate
         W3(1) = CTE3*W2(1)-STE3*W2(2)+L113*Dom2(1)+L213*Dom2(2)+L313*Dom
     $ 2(3)+Dq(3)*L123*om2(1)+Dq(3)*L223*om2(2)+Dq(3)*L323*om2(3)
c        *  2nd coordinate
         W3(2) = STE3*W2(1)+CTE3*W2(2)+L123*Dom2(1)+L223*Dom2(2)+L323*Dom
     $ 2(3)-Dq(3)*L113*om2(1)-Dq(3)*L213*om2(2)-Dq(3)*L313*om2(3)
c        *  3rd coordinate
         W3(3) = L233*Dom2(2)
c
c                 --------- The 4th link ---------
c
c        *  1st coordinate
         W4(1) = CTE4*W3(1)-STE4*W3(3)+L114*Dom3(1)+L214*Dom3(2)+L314*Dom
     $ 3(3)+Dq(4)*L124*om3(1)+Dq(4)*L224*om3(2)+Dq(4)*L324*om3(3)+Dq(4)*
     $ STE4*vl3(1)+Dq(4)*CTE4*vl3(3)
c        *  2nd coordinate
         W4(2) = STE4*W3(1)+CTE4*W3(3)+L124*Dom3(1)+L224*Dom3(2)+L324*Dom
     $ 3(3)-Dq(4)*L114*om3(1)-Dq(4)*L214*om3(2)-Dq(4)*L314*om3(3)-Dq(4)*
     $ CTE4*vl3(1)+Dq(4)*STE4*vl3(3)
c        *  3rd coordinate
         W4(3) = -W3(2)+L334*Dom3(3)
c
c                 --------- The 5th link ---------
c
c        *  1st coordinate
         W5(1) = CTE5*W4(1)+STE5*W4(3)+Dq(5)*STE5*vl4(1)-Dq(5)*CTE5*vl4(3
     $ )
c        *  2nd coordinate
         W5(2) = STE5*W4(1)-CTE5*W4(3)-Dq(5)*CTE5*vl4(1)-Dq(5)*STE5*vl4(3
     $ )
c        *  3rd coordinate
         W5(3) = W4(2)
c
c                 --------- The 6th link ---------
c
c        *  1st coordinate
```

```
        W6(1) = CTE6*W5(1)-STE6*W5(3)+Dq(6)*STE6*vl5(1)+Dq(6)*CTE6*vl5(3
     $ )
c         * 2nd coordinate
        W6(2) = STE6*W5(1)+CTE6*W5(3)-Dq(6)*CTE6*vl5(1)+Dq(6)*STE6*vl5(3
     $ )
c         * 3rd coordinate
        W6(3) = -W5(2)
c
c    -- Now the program computes inertial forces acting at
c    the center of mass.
c
c
c                --------- The 1st link ---------
c
c
c   Local priliminary calculations
c
       VOG1(1) = -om1(3)*ros1(2)
       VOG1(2) = om1(3)*ros1(1)
c    VOG1(3) is equal to zero
c
c         * 1st coordinate
      Fe1(1) = -m1*(Dom1(3)*ros1(2)+om1(3)*VOG1(2))
c         * 2nd coordinate
      Fe1(2) = m1*(Dom1(3)*ros1(1)+om1(3)*VOG1(1))
c         * 3rd coordinate
      Fe1(3) = m1*W1(3)
c
c                --------- The 2nd link ---------
c
c
c   Local priliminary calculations
c
       VOG2(1) = om2(2)*ros2(3)-om2(3)*ros2(2)
       VOG2(2) = om2(3)*ros2(1)-om2(1)*ros2(3)
       VOG2(3) = om2(1)*ros2(2)-om2(2)*ros2(1)
c         * 1st coordinate
      Fe2(1) = -m2*(-W2(1)-Dom2(2)*ros2(3)+Dom2(3)*ros2(2)-om2(2)*VOG2(3
     $ )+om2(3)*VOG2(2))
c         * 2nd coordinate
      Fe2(2) = m2*(W2(2)+Dom2(3)*ros2(1)-Dom2(1)*ros2(3)+om2(3)*VOG2(1)-
     $ om2(1)*VOG2(3))
c         * 3rd coordinate
      Fe2(3) = m2*(Dom2(1)*ros2(2)-Dom2(2)*ros2(1)+om2(1)*VOG2(2)-om2(2)
     $ *VOG2(1))
c
c                --------- The 3rd link ---------
c
c
c   Local priliminary calculations
c
       VOG3(1) = vl3(1)+om3(2)*ros3(3)-om3(3)*ros3(2)
```

```
      VOG3(2) = vl3(2)+om3(3)*ros3(1)-om3(1)*ros3(3)
      VOG3(3) = vl3(3)+om3(1)*ros3(2)-om3(2)*ros3(1)
c       * 1st coordinate
     Fe3(1) = -m3*(-W3(1)-Dom3(2)*ros3(3)+Dom3(3)*ros3(2)-om3(2)*VOG3(3
    $ )+om3(3)*VOG3(2))
c       * 2nd coordinate
     Fe3(2) = m3*(W3(2)+Dom3(3)*ros3(1)-Dom3(1)*ros3(3)+om3(3)*VOG3(1)-
    $ om3(1)*VOG3(3))
c       * 3rd coordinate
     Fe3(3) = -m3*(-W3(3)-Dom3(1)*ros3(2)+Dom3(2)*ros3(1)-om3(1)*VOG3(2
    $ )+om3(2)*VOG3(1))
c
c               --------- The 4th link ---------
c
c
c   Local priliminary calculations
c
      VOG4(1) = vl4(1)+om4(2)*ros4(3)-om4(3)*ros4(2)
      VOG4(2) = vl4(2)+om4(3)*ros4(1)-om4(1)*ros4(3)
      VOG4(3) = vl4(3)+om4(1)*ros4(2)-om4(2)*ros4(1)
c       * 1st coordinate
     Fe4(1) = m4*(W4(1)+Dom4(2)*ros4(3)-Dom4(3)*ros4(2)+om4(2)*VOG4(3)-
    $ om4(3)*VOG4(2))
c       * 2nd coordinate
     Fe4(2) = m4*(W4(2)+Dom4(3)*ros4(1)-Dom4(1)*ros4(3)+om4(3)*VOG4(1)-
    $ om4(1)*VOG4(3))
c       * 3rd coordinate
     Fe4(3) = m4*(W4(3)+Dom4(1)*ros4(2)-Dom4(2)*ros4(1)+om4(1)*VOG4(2)-
    $ om4(2)*VOG4(1))
c
c               --------- The 5th link ---------
c
c
c   Local priliminary calculations
c
      VOG5(1) = vl5(1)+om5(2)*ros5(3)-om5(3)*ros5(2)
      VOG5(2) = vl5(2)+om5(3)*ros5(1)-om5(1)*ros5(3)
      VOG5(3) = vl5(3)+om5(1)*ros5(2)-om5(2)*ros5(1)
c       * 1st coordinate
     Fe5(1) = m5*(W5(1)+Dom5(2)*ros5(3)-Dom5(3)*ros5(2)+om5(2)*VOG5(3)-
    $ om5(3)*VOG5(2))
c       * 2nd coordinate
     Fe5(2) = -m5*(-W5(2)-Dom5(3)*ros5(1)+Dom5(1)*ros5(3)-om5(3)*VOG5(1
    $ )+om5(1)*VOG5(3))
c       * 3rd coordinate
     Fe5(3) = m5*(W5(3)+Dom5(1)*ros5(2)-Dom5(2)*ros5(1)+om5(1)*VOG5(2)-
    $ om5(2)*VOG5(1))
c
c               --------- The 6th link ---------
c
c
c   Local priliminary calculations
```

```
c
       VOG6(1) = vl6(1)+om6(2)*ros6(3)-om6(3)*ros6(2)
       VOG6(2) = vl6(2)+om6(3)*ros6(1)-om6(1)*ros6(3)
       VOG6(3) = vl6(3)+om6(1)*ros6(2)-om6(2)*ros6(1)
c        *  1st coordinate
      Fe6(1) = m6*(W6(1)+Dom6(2)*ros6(3)-Dom6(3)*ros6(2)+om6(2)*VOG6(3)-
     $ om6(3)*VOG6(2))
c        *  2nd coordinate
      Fe6(2) = m6*(W6(2)+Dom6(3)*ros6(1)-Dom6(1)*ros6(3)+om6(3)*VOG6(1)-
     $ om6(1)*VOG6(3))
c        *  3rd coordinate
      Fe6(3) = -m6*(-W6(3)-Dom6(1)*ros6(2)+Dom6(2)*ros6(1)-om6(1)*VOG6(2
     $ )+om6(2)*VOG6(1))
c
c    -- Now the program computes moments in the origine of
c    frame "i" resulting of the inertial motion.
c
c
c                --------- The 1st link ---------
c
c
c   Local priliminary calculations
c
       IOM1(1) = -Ixz(1)*om1(3)
       IOM1(2) = -Iyz(1)*om1(3)
       IOM1(3) = Izz(1)*om1(3)
c        *  1st coordinate
       Ne1(1) = -Ixz(1)*Dom1(3)-om1(3)*IOM1(2)+ros1(2)*Fe1(3)-ros1(3)*F
     $ e1(2)
c        *  2nd coordinate
       Ne1(2) = -Iyz(1)*Dom1(3)+om1(3)*IOM1(1)+ros1(3)*Fe1(1)-ros1(1)*F
     $ e1(3)
c        *  3rd coordinate
       Ne1(3) = Izz(1)*Dom1(3)+ros1(1)*Fe1(2)-ros1(2)*Fe1(1)
c
c                --------- The 2nd link ---------
c
c
c   Local priliminary calculations
c
       IOM2(1) = Ixx(2)*om2(1)-Ixy(2)*om2(2)-Ixz(2)*om2(3)
       IOM2(2) = -Ixy(2)*om2(1)+Iyy(2)*om2(2)-Iyz(2)*om2(3)
       IOM2(3) = -Ixz(2)*om2(1)-Iyz(2)*om2(2)+Izz(2)*om2(3)
c        *  1st coordinate
       Ne2(1) = Ixx(2)*Dom2(1)-Ixy(2)*Dom2(2)-Ixz(2)*Dom2(3)+om2(2)*IOM
     $ 2(3)-om2(3)*IOM2(2)+ros2(2)*Fe2(3)-ros2(3)*Fe2(2)
c        *  2nd coordinate
       Ne2(2) = -Ixy(2)*Dom2(1)+Iyy(2)*Dom2(2)-Iyz(2)*Dom2(3)+om2(3)*IO
     $ M2(1)-om2(1)*IOM2(3)+ros2(3)*Fe2(1)-ros2(1)*Fe2(3)
c        *  3rd coordinate
       Ne2(3) = -Ixz(2)*Dom2(1)-Iyz(2)*Dom2(2)+Izz(2)*Dom2(3)+om2(1)*IO
     $ M2(2)-om2(2)*IOM2(1)+ros2(1)*Fe2(2)-ros2(2)*Fe2(1)
```

```
c
c                --------- The 3rd link ---------
c
c
c   Local priliminary calculations
c
       IOM3(1) = Ixx(3)*om3(1)-Ixy(3)*om3(2)-Ixz(3)*om3(3)
       IOM3(2) = -Ixy(3)*om3(1)+Iyy(3)*om3(2)-Iyz(3)*om3(3)
       IOM3(3) = -Ixz(3)*om3(1)-Iyz(3)*om3(2)+Izz(3)*om3(3)
c        * 1st coordinate
        Ne3(1) = Ixx(3)*Dom3(1)-Ixy(3)*Dom3(2)-Ixz(3)*Dom3(3)+om3(2)*IOM
     $ 3(3)-om3(3)*IOM3(2)+ros3(2)*Fe3(3)-ros3(3)*Fe3(2)
c        * 2nd coordinate
        Ne3(2) = -Ixy(3)*Dom3(1)+Iyy(3)*Dom3(2)-Iyz(3)*Dom3(3)+om3(3)*IO
     $ M3(1)-om3(1)*IOM3(3)+ros3(3)*Fe3(1)-ros3(1)*Fe3(3)
c        * 3rd coordinate
        Ne3(3) = -Ixz(3)*Dom3(1)-Iyz(3)*Dom3(2)+Izz(3)*Dom3(3)+om3(1)*IO
     $ M3(2)-om3(2)*IOM3(1)+ros3(1)*Fe3(2)-ros3(2)*Fe3(1)
c
c                --------- The 4th link ---------
c
c
c   Local priliminary calculations
c
       IOM4(1) = Ixx(4)*om4(1)-Ixy(4)*om4(2)-Ixz(4)*om4(3)
       IOM4(2) = -Ixy(4)*om4(1)+Iyy(4)*om4(2)-Iyz(4)*om4(3)
       IOM4(3) = -Ixz(4)*om4(1)-Iyz(4)*om4(2)+Izz(4)*om4(3)
c        * 1st coordinate
        Ne4(1) = Ixx(4)*Dom4(1)-Ixy(4)*Dom4(2)-Ixz(4)*Dom4(3)+om4(2)*IOM
     $ 4(3)-om4(3)*IOM4(2)+ros4(2)*Fe4(3)-ros4(3)*Fe4(2)
c        * 2nd coordinate
        Ne4(2) = -Ixy(4)*Dom4(1)+Iyy(4)*Dom4(2)-Iyz(4)*Dom4(3)+om4(3)*IO
     $ M4(1)-om4(1)*IOM4(3)+ros4(3)*Fe4(1)-ros4(1)*Fe4(3)
c        * 3rd coordinate
        Ne4(3) = -Ixz(4)*Dom4(1)-Iyz(4)*Dom4(2)+Izz(4)*Dom4(3)+om4(1)*IO
     $ M4(2)-om4(2)*IOM4(1)+ros4(1)*Fe4(2)-ros4(2)*Fe4(1)
c
c                --------- The 5th link ---------
c
c
c   Local priliminary calculations
c
       IOM5(1) = Ixx(5)*om5(1)-Ixy(5)*om5(2)-Ixz(5)*om5(3)
       IOM5(2) = -Ixy(5)*om5(1)+Iyy(5)*om5(2)-Iyz(5)*om5(3)
       IOM5(3) = -Ixz(5)*om5(1)-Iyz(5)*om5(2)+Izz(5)*om5(3)
c        * 1st coordinate
        Ne5(1) = Ixx(5)*Dom5(1)-Ixy(5)*Dom5(2)-Ixz(5)*Dom5(3)+om5(2)*IOM
     $ 5(3)-om5(3)*IOM5(2)+ros5(2)*Fe5(3)-ros5(3)*Fe5(2)
c        * 2nd coordinate
        Ne5(2) = -Ixy(5)*Dom5(1)+Iyy(5)*Dom5(2)-Iyz(5)*Dom5(3)+om5(3)*IO
     $ M5(1)-om5(1)*IOM5(3)+ros5(3)*Fe5(1)-ros5(1)*Fe5(3)
c        * 3rd coordinate
```

```
        Ne5(3) = -Ixz(5)*Dom5(1)-Iyz(5)*Dom5(2)+Izz(5)*Dom5(3)+om5(1)*IO
     $ M5(2)-om5(2)*IOM5(1)+ros5(1)*Fe5(2)-ros5(2)*Fe5(1)
c
c               --------- The 6th link ---------
c
c
c   Local priliminary calculations
c
        IOM6(1) = Ixx(6)*om6(1)-Ixy(6)*om6(2)-Ixz(6)*om6(3)
        IOM6(2) = -Ixy(6)*om6(1)+Iyy(6)*om6(2)-Iyz(6)*om6(3)
        IOM6(3) = -Ixz(6)*om6(1)-Iyz(6)*om6(2)+Izz(6)*om6(3)
c         * 1st coordinate
        Ne6(1) = Ixx(6)*Dom6(1)-Ixy(6)*Dom6(2)-Ixz(6)*Dom6(3)+om6(2)*IOM
     $ 6(3)-om6(3)*IOM6(2)+ros6(2)*Fe6(3)-ros6(3)*Fe6(2)
c         * 2nd coordinate
        Ne6(2) = -Ixy(6)*Dom6(1)+Iyy(6)*Dom6(2)-Iyz(6)*Dom6(3)+om6(3)*IO
     $ M6(1)-om6(1)*IOM6(3)+ros6(3)*Fe6(1)-ros6(1)*Fe6(3)
c         * 3rd coordinate
        Ne6(3) = -Ixz(6)*Dom6(1)-Iyz(6)*Dom6(2)+Izz(6)*Dom6(3)+om6(1)*IO
     $ M6(2)-om6(2)*IOM6(1)+ros6(1)*Fe6(2)-ros6(2)*Fe6(1)
c
c    -- Now the program computes forces exerted by link i-1
c   on link i.
c
c
c               --------- The 6th link ---------
c
c         * 1st coordinate
        fa6(1) = Fe6(1)
c         * 2nd coordinate
        fa6(2) = Fe6(2)
c         * 3rd coordinate
        fa6(3) = Fe6(3)
c
c               --------- The 5th link ---------
c
c         * 1st coordinate
        fa5(1) = CTE6*fa6(1)+STE6*fa6(2)+Fe5(1)
c         * 2nd coordinate
        fa5(2) = -fa6(3)+Fe5(2)
c         * 3rd coordinate
        fa5(3) = -STE6*fa6(1)+CTE6*fa6(2)+Fe5(3)
c
c               --------- The 4th link ---------
c
c         * 1st coordinate
        fa4(1) = CTE5*fa5(1)+STE5*fa5(2)+Fe4(1)
c         * 2nd coordinate
        fa4(2) = fa5(3)+Fe4(2)
c         * 3rd coordinate
        fa4(3) = STE5*fa5(1)-CTE5*fa5(2)+Fe4(3)
c
```

```
c                --------- The 3rd link ---------
c
c        *  1st coordinate
         fa3(1) = CTE4*fa4(1)+STE4*fa4(2)+Fe3(1)
c        *  2nd coordinate
         fa3(2) = -fa4(3)+Fe3(2)
c        *  3rd coordinate
         fa3(3) = -STE4*fa4(1)+CTE4*fa4(2)+Fe3(3)
c
c                --------- The 2nd link ---------
c
c        *  1st coordinate
         fa2(1) = CTE3*fa3(1)+STE3*fa3(2)+Fe2(1)
c        *  2nd coordinate
         fa2(2) = -STE3*fa3(1)+CTE3*fa3(2)+Fe2(2)
c        *  3rd coordinate
         fa2(3) = fa3(3)+Fe2(3)
c
c                --------- The 1st link ---------
c
c        *  1st coordinate
         fa1(1) = CTE2*fa2(1)+STE2*fa2(2)+Fe1(1)
c        *  2nd coordinate
         fa1(2) = -fa2(3)+Fe1(2)
c        *  3rd coordinate
         fa1(3) = -STE2*fa2(1)+CTE2*fa2(2)+Fe1(3)
c
c     -- Now the program computes torque exerted by link i-1
c    on link i.
c
c
c                --------- The 6th link ---------
c
c        *  1st coordinate
         na6(1) = Ne6(1)
c        *  2nd coordinate
         na6(2) = Ne6(2)
c        *  3rd coordinate
         na6(3) = Ne6(3)
c
c                --------- The 5th link ---------
c
c
c    Local priliminary calculations
c
        NFR5(1) = na6(1)
        NFR5(2) = na6(2)
        NFR5(3) = na6(3)
c        *  1st coordinate
         na5(1) = Ne5(1)+CTE6*NFR5(1)+STE6*NFR5(2)
c        *  2nd coordinate
         na5(2) = Ne5(2)-NFR5(3)
```

```
c         * 3rd coordinate
          na5(3) = Ne5(3)-STE6*NFR5(1)+CTE6*NFR5(2)
c
c                 --------- The 4th link ---------
c
c
c   Local priliminary calculations
c
          NFR4(1) = na5(1)
          NFR4(2) = na5(2)
          NFR4(3) = na5(3)
c         * 1st coordinate
          na4(1) = Ne4(1)+CTE5*NFR4(1)+STE5*NFR4(2)
c         * 2nd coordinate
          na4(2) = Ne4(2)+NFR4(3)
c         * 3rd coordinate
          na4(3) = Ne4(3)+STE5*NFR4(1)-CTE5*NFR4(2)
c
c                 --------- The 3rd link ---------
c
c
c   Local priliminary calculations
c
          NFR3(1) = na4(1)+L114*fa4(1)+L214*fa4(2)+L314*fa4(3)
          NFR3(2) = na4(2)+L124*fa4(1)+L224*fa4(2)+L324*fa4(3)
          NFR3(3) = na4(3)+L334*fa4(3)
c         * 1st coordinate
          na3(1) = Ne3(1)+CTE4*NFR3(1)+STE4*NFR3(2)
c         * 2nd coordinate
          na3(2) = Ne3(2)-NFR3(3)
c         * 3rd coordinate
          na3(3) = Ne3(3)-STE4*NFR3(1)+CTE4*NFR3(2)
c
c                 --------- The 2nd link ---------
c
c
c   Local priliminary calculations
c
          NFR2(1) = na3(1)+L113*fa3(1)+L213*fa3(2)+L313*fa3(3)
          NFR2(2) = na3(2)+L123*fa3(1)+L223*fa3(2)+L323*fa3(3)
          NFR2(3) = na3(3)+L233*fa3(2)
c         * 1st coordinate
          na2(1) = Ne2(1)+CTE3*NFR2(1)+STE3*NFR2(2)
c         * 2nd coordinate
          na2(2) = Ne2(2)-STE3*NFR2(1)+CTE3*NFR2(2)
c         * 3rd coordinate
          na2(3) = Ne2(3)+NFR2(3)
c
c                 --------- The 1st link ---------
c
c
c   Local priliminary calculations
```

```
c
      NFR1(1) = na2(1)
      NFR1(2) = na2(2)
      NFR1(3) = na2(3)
c       * 1st coordinate
       na1(1) = Ne1(1)+CTE2*NFR1(1)+STE2*NFR1(2)
c       * 2nd coordinate
       na1(2) = Ne1(2)-NFR1(3)
c       * 3rd coordinate
       na1(3) = Ne1(3)-STE2*NFR1(1)+CTE2*NFR1(2)
c
c    -- The generalized torque in the 1st joint.
c
      Qg(1) = na1(3)
c
c    -- The generalized torque in the 2nd joint.
c
      Qg(2) = na2(3)
c
c    -- The generalized torque in the 3rd joint.
c
      Qg(3) = na3(3)
c
c    -- The generalized torque in the 4th joint.
c
      Qg(4) = na4(3)
c
c    -- The generalized torque in the 5th joint.
c
      Qg(5) = na5(3)
c
c    -- The generalized torque in the 6th joint.
c
      Qg(6) = na6(3)
     return
c    -- All's well that ends well
     end
```

Annexe E

COMPLEXITES EFFECTIVES APRES COMPILATION DU CODE EN LIGNE

E.1 Introduction

Revenons aux exemples donnés dans le paragraphe 6.3.1 [page 176]. Les compilateurs Fortran sont réputés pour leur optimiseurs. Le but de l'annexe présent est de prouver que, même après compilation, les codes générés par la primitive `fortran` de Maple avec l'option `optimized` ne donnent pas nécessairement de complexités optimales.

E.2 Code assembleur de l'implémentation initiale

Le code en assembleur, généré par le compilateur Fortran 77, associé au code en ligne:

```
t4 = b1*c2-b2*c1
t9 = b3*c1-b1*c3
t16 = b2*c3-b3*c2
d(1) = a2*t4-a3*t9
d(2) = a3*t16-a1*t4
d(3) = a1*t9-a2*t16
```

est:

```
.seg "data"
.seg "data"
.seg "bss"
.seg "data"
.align 8
.seg "bss"
.align 8
.reserve VAR_SEG1,32,"bss"
.seg "text"
.proc 020
```

```
.global _ali_

_ali_:
!#PROLOGUE# 0
sethi %hi(LF1),%g1
add %g1,%lo(LF1),%g1
save %sp,%g1,%sp
!#PROLOGUE# 1
st %i1,[%fp+0x48]
st %i2,[%fp+0x4c]
st %i3,[%fp+0x50]
st %i5,[%fp+0x58]
L16:
sethi %hi(VAR_SEG1+4096),%l7
add %l7,%lo(VAR_SEG1+4096),%l7
L14:
ld [%fp+0x58],%o0
ld2 [%o0],%f0
ld [%fp+0x4c],%o1
ld2 [%o1],%f2
fmuld %f2,%f0,%f4
ld [%fp+0x60],%o2
ld2 [%o2],%f6
ld [%fp+0x48],%o3
ld2 [%o3],%f8
fmuld %f8,%f6,%f10
fsubd %f10,%f4,%f12
ld [%fp+0x64],%o4
st2 %f12,[%o4]
ld [%fp+0x5c],%o5
ld2 [%o5],%f14
ld [%fp+0x48],%o7
ld2 [%o7],%f16
fmuld %f16,%f14,%f18
ld [%fp+0x58],%l0
ld2 [%l0],%f20
ld [%fp+0x50],%l1
ld2 [%l1],%f22
fmuld %f22,%f20,%f24
fsubd %f24,%f18,%f26
ld [%fp+0x68],%l2
st2 %f26,[%l2]
ld [%fp+0x60],%l3
ld2 [%l3],%f28
ld [%fp+0x50],%l4
ld2 [%l4],%f30
fmuld %f30,%f28,%f0
ld [%fp+0x5c],%l5
ld2 [%l5],%f2
ld [%fp+0x4c],%l6
ld2 [%l6],%f4
fmuld %f4,%f2,%f6
```

```
fsubd %f6,%f0,%f8
st2 %f8,[%l7+-0xfe8]
ld [%fp+0x68],%i0
ld2 [%i0],%f10
ld2 [%l7+-0xff0],%f12
fmuld %f12,%f10,%f14
ld [%fp+0x64],%i1
ld2 [%i1],%f16
ld2 [%l7+-0xff8],%f18
fmuld %f18,%f16,%f20
fsubd %f20,%f14,%f22
ld [%fp+0x74],%i2
st2 %f22,[%i2]
ld [%fp+0x64],%i3
ld2 [%i3],%f24
ld2 [%l7+-0x1000],%f26
fmuld %f26,%f24,%f28
ld2 [%l7+-0xff0],%f30
ld2 [%l7+-0xfe8],%f0
fmuld %f30,%f0,%f2
fsubd %f2,%f28,%f4
ld [%fp+0x7c],%i4
st2 %f4,[%i4]
ld [%fp+0x68],%i5
ld2 [%i5],%f6
ld2 [%l7+-0x1000],%f8
fmuld %f8,%f6,%f10
ld2 [%l7+-0xff8],%f12
ld2 [%l7+-0xfe8],%f14
fmuld %f12,%f14,%f16
fsubd %f10,%f16,%f18
ld [%fp+0x78],%o0
st2 %f18,[%o0]
b L13
nop
L13:
L11:
L12:
b LE1
nop
LE1:
ret
restore
.optim "-O~Q~R~S"
       LF1 = -64
LP1 = 64
LST1 = 64
LT1 = 64
.seg "data"
```

donc, on a bien 12 multiplications (12 occurrences de "`fmuld`" dans le code assembleur) et 6 additions (6 occurrences de "`fsubd`" dans le code assembleur; en réalité, c'est des soustractions).

E.3 Code assembleur de l'implémentation transformée

De même, pour le code en ligne obtenu par la traduction du double produit vectoriel après sa transformation par la fonction `simplify`:

```
t1 = b1*c2
t3 = b2*c1
t6 = b3*c1
t9 = b1*c3
t12 = b2*c3
t14 = b3*c2
d(1) = a2*t1-a2*t3-a3*t6+a3*t9
d(2) = a3*t12-a3*t14-a1*t1+a1*t3
d(3) = a1*t6-a1*t9-a2*t12+a2*t14
```

le code en assembleur généré par le compilateur Fortran 77, est:

```
.seg "data"
.seg "data"
.seg "bss"
.seg "data"
.align 8
.seg "bss"
.align 8
.reserve VAR_SEG1,24,"bss"
.seg "text"
.proc 020
.global _ali_

_ali_:
!#PROLOGUE# 0
sethi %hi(LF1),%g1
add %g1,%lo(LF1),%g1
save %sp,%g1,%sp
!#PROLOGUE# 1
st %i0,[%fp+0x44]
st %i1,[%fp+0x48]
st %i2,[%fp+0x4c]
st %i3,[%fp+0x50]
st %i4,[%fp+0x54]
st %i5,[%fp+0x58]
L16:
sethi %hi(VAR_SEG1+4096),%l7
add %l7,%lo(VAR_SEG1+4096),%l7
L14:
ld [%fp+0x60],%o0
ld2 [%o0],%f0
ld [%fp+0x48],%o1
ld2 [%o1],%f2
fmuld %f2,%f0,%f4
ld [%fp+0x44],%o2
st2 %f4,[%o2]
```

```
ld [%fp+0x58],%o3
ld2 [%o3],%f6
ld [%fp+0x4c],%o4
ld2 [%o4],%f8
fmuld %f8,%f6,%f10
ld [%fp+0x54],%o5
st2 %f10,[%o5]
ld [%fp+0x58],%o7
ld2 [%o7],%f12
ld [%fp+0x50],%l0
ld2 [%l0],%f14
fmuld %f14,%f12,%f16
ld [%fp+0x64],%l1
st2 %f16,[%l1]
ld [%fp+0x5c],%l2
ld2 [%l2],%f18
ld [%fp+0x48],%l3
ld2 [%l3],%f20
fmuld %f20,%f18,%f22
ld [%fp+0x68],%l4
st2 %f22,[%l4]
ld [%fp+0x5c],%l5
ld2 [%l5],%f24
ld [%fp+0x4c],%l6
ld2 [%l6],%f26
fmuld %f26,%f24,%f28
ld [%fp+0x6c],%i0
st2 %f28,[%i0]
ld [%fp+0x60],%i1
ld2 [%i1],%f30
ld [%fp+0x50],%i2
ld2 [%i2],%f0
fmuld %f0,%f30,%f2
ld [%fp+0x70],%i3
st2 %f2,[%i3]
ld [%fp+0x54],%i4
ld2 [%i4],%f4
ld2 [%l7+-0xff8],%f6
fmuld %f6,%f4,%f8
ld [%fp+0x44],%i5
ld2 [%i5],%f10
ld2 [%l7+-0xff8],%f12
fmuld %f12,%f10,%f14
fsubd %f14,%f8,%f16
ld [%fp+0x64],%o0
ld2 [%o0],%f18
ld2 [%l7+-0xff0],%f20
fmuld %f20,%f18,%f22
fsubd %f16,%f22,%f24
ld [%fp+0x68],%o1
ld2 [%o1],%f26
ld2 [%l7+-0xff0],%f28
```

```
fmuld %f28,%f26,%f30
faddd %f24,%f30,%f0
ld [%fp+0x74],%o2
st2 %f0,[%o2]
ld [%fp+0x70],%o3
ld2 [%o3],%f2
ld2 [%l7+-0xff0],%f4
fmuld %f4,%f2,%f6
ld [%fp+0x6c],%o4
ld2 [%o4],%f8
ld2 [%l7+-0xff0],%f10
fmuld %f10,%f8,%f12
fsubd %f12,%f6,%f14
ld [%fp+0x44],%o5
ld2 [%o5],%f16
ld2 [%l7+-0x1000],%f18
fmuld %f18,%f16,%f20
fsubd %f14,%f20,%f22
ld [%fp+0x54],%o7
ld2 [%o7],%f24
ld2 [%l7+-0x1000],%f26
fmuld %f26,%f24,%f28
faddd %f22,%f28,%f30
ld [%fp+0x7c],%l0
st2 %f30,[%l0]
ld [%fp+0x68],%l1
ld2 [%l1],%f0
ld2 [%l7+-0x1000],%f2
fmuld %f2,%f0,%f4
ld [%fp+0x64],%l2
ld2 [%l2],%f6
ld2 [%l7+-0x1000],%f8
fmuld %f8,%f6,%f10
fsubd %f10,%f4,%f12
ld [%fp+0x6c],%l3
ld2 [%l3],%f14
ld2 [%l7+-0xff8],%f16
fmuld %f16,%f14,%f18
fsubd %f12,%f18,%f20
ld [%fp+0x70],%l4
ld2 [%l4],%f22
ld2 [%l7+-0xff8],%f24
fmuld %f24,%f22,%f26
faddd %f20,%f26,%f28
ld [%fp+0x78],%l5
st2 %f28,[%l5]
b L13
nop
L13:
L11:
L12:
b LE1
```

```
nop
LE1:
ret
restore
.optim "-O~Q~R~S"
       LF1 = -64
LP1 = 64
LST1 = 64
LT1 = 64
.seg "data"
```

le calcul est effectué en 9 additions et 18 multiplications et cela donne absolument le même résultat, même si on utilise l'option d'optimisation "-O" du compilateur Fortran.

E.4 Conclusion

Cela confirme notre critique de l'optimiseur du code Fortran (sous Maple) des expressions formelles. La compilation des deux implémentations ne donne pas la même complexité. La complexité du code obtenu après une transformation formelle n'est pas nécessairement optimale.

REFERENCES ET BIBLIOGRAPHIE

[ABEL-87] H. Abelson, G.J. Sussman & J. Sussman "Structure and interpretation of computer programs", MIT Press, McGraw-Hill, (1987).

[ABRA-78] R. Abraham & J.E. Marsden "Fundations of Mechanics", The Benjamin/Cummings Publishing Compagny, (1978).

[ALDO-82] M.G. Aldon "Elaboration Automatique de Modèles Dynamiques de Robots en Vue de leur Conception et leur Commande", Thèse de 3^e cycle, Université des Sciences et Thechniques du Languedoc, (1982).

[ANGE-82] J. Angeles "Spatial Kinematic Chains", Springer-Verlag, (1982).

[ANGE-88] J. Angeles "Rational Kinematics", Springer-Verlag, (1988).

[ANGE-90] J. Angeles, A. Alivizatos & P.J. Zsombor-Muray "The Synthesis of Smooth Trajectories for Pick and Place Operations", IEEE Trans. on Syst., Man and Cybernetics, Vol. 18, pp. 173-178, (1990).

[APPE-00] P. Appell "Sur la Forme Générale des Equations de Dynamique", J. für die Reine und Angewandte Mathematique. Vol. 121, pp. 310–319, (1900).

[APPE-11] P. Appell "Traité de Mécanique Rationnelle", Tome 2, 3^e édition, Paris, (1911).

[ARNO-66] V.I. Arnold "Sur la Géométrie Différentielle des Groupes de Lie de Dimension Infinie et ses Applications à l'Hydrodynamique des Fluides Parfaits", Annales de l'Inst. Fourier, Grenoble, Vol. 16(1), pp. 319–361, (1966).

[ARNO-78] V.I. Arnold "Mathematical Methods of Classical Mechanics", Springer-Verlag, (1978).

[BARR-81] A. Barr and E.A. Feingenbaum "The Handbook of Artificial Intelligence", Pitman, London, (1981).

[BEJE-74] A.K. Bejeczy "Robot Arm Dynamic and Control", NASA Tech. Memo. n^o 33-669. JPL, Feb. (1974).

[BERT-10] J. Bertrand "Dialogo Sopra le due Massimi Sistemi del Mundo", Edition Florence, (1710).

[BOUR-70] N. Bourbaki "Eléments de Mathématique: Algèbre", Chapitres 1 à 3, Diffusion CCLS, (1970).

[CHAR-91] B.W. Char & al. "Maple V Language Reference Manual", Springer-Verlag, (1991).

[CHAR-91] B.W. Char & al. "Maple V Library Reference Manual", Springer-Verlag, (1991).

[CHAR-92] B.W. Char & al. "First Leaves: A Tutorial Introduction to Maple V", Springer-Verlag, (1992).

[CHEV-84] D. Chevallier & Helmer "Formation des Equations de la Dynamique: Examen des Divers Méthodes", Annales des Ponts et Chaussées , Paris, 1er Trimestre, Vol.29, pp. 5–19, (1984).

[CHEV-86] D. Chevallier "Groupes de Lie et Mécanique des Systèmes de Corps Rigides", Mathematishe Modellierung ein Arbeitsbuch für Seminare, Mac-Graw Hill, pp. 231-269,4 Oct. (1986).

[CHEV-91] D. Chevallier "Lie Algebras, Modules, Dual Quaternions and Algebraic Methods in Kinematics", Mechanisms and Machine Theory, vol. 26(6), pp. 613-627, (1991).

[CHUR-56] A. Church "Introduction to Mathematical Logic", Vol. 1, Prinston University Press, (1956).

[CLI-873] W.K. Clifford "Preliminary Sketch of Biquaternions", Jour. of Math. Pure and Applied, Vol. 4, (1873).

[CLI-878] W.K. Clifford "Applications of Grassmann's Extensive Algebra", Jour. of Math. Pure and Applied, Volume 1 pp. 350-358, (1878).

[CLOC-84] W.F. Clocksin & C.S. Mellish "Programming in Prolog", 2nd ed., Springer Verlag, (1984).

[COIF-81] P. Coiffet "Les Robots: Modélisation et Commande", Tome 1, Hermès, (1981).

[DENA-55] J. Denavit & R.S. Hartenberg "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", ASME Journal of Applied Mechanics, pp. 215-221, Jun. (1955).

[DERS-89] N. Dershowitz & J.P. Jouammaud "Rewrite Systems", In J.V. Leeuwen, Handbook of Theoretical Computer Science, Chap. 15, North-Holland, Amsterdam, (1989).

[DESL-87] E.A. Desloge "Relationship Between Kane's Equations and the Gibbs–Appel Equations", J. of Guidance and Control Dynamics, Vol. 12, (1987).

[DIE—71] J. Dieudonné "Eléments d'Analyse", Tome 4, Cahiers Scientifiques, Fascicule XXXIV, Gautier-Villars, (1971).

[DIE—75] J. Dieudonné "Eléments d'Analyse", Tome 5, Cahiers Scientifiques, Fascicule XXXVIII, Gautier-Villars, (1975).

[DIME-65] F.M. Dimentberg "Vintovoye Ischislenye i Yego Prilozenia V Mekhanike" [The Screw Calculus and its Applications in Mechanics], **Издател'ство "Наука"**, (1965).

[DOMB-88] E. Dombre & W. Khalil "Modélisation et Commande des Robots", Hermés, (1988).

[ESPI-86] B. Espiau "An Overview of Local Environment Sensing in Robotics Applications", NATO, Advenced Research Workschop, Sensors and Sensory Systems for Advenced Robotics, Springer-Verlag ed., Maratea, Italia, April (1986).

[FARR-87] H. Farreny et M. Ghallab "Eléments d'Intelligence Artificielle", Hermes, (1987).

[FAVA-57] J. Favard "Cours de Géométrie Différentielle Locale", Gautier-Villars, (1957).

[FLAJ-90] P. Flajolet "On Adaptative Sampling", Computing, Vol. 34, pp. 391-400, (1990).

[FRAN-69] J. Frank Adams "Lectures on Lie Groups", The University of Chicago Press, (1969).

[FIES-84] D. Fieschi "Contribution au Système Expert Sphinx: Appalication à l'Enseignement Médical", Thèse de 3^e Cycle, Université Paris VI, (1984).

[FISC-06] O. Fischer "Einführung in die Mechanik Lebender Mechanismen", Leipzig, (1906).

[GARN-90] Ch. Garnier "Une Application en Maple: le Logiciel Gemmes", V^e Ecole Europeenne sur l'Utilisation du Calcul Formel en Ingénierie, INRIA-Sophia-Antipolis, (1990).

[GERA-89] M. Geradin & Cardona "Time Integration of the Equation of Motion in Mechanism Analysis", Computer and Structurs, Vol. 33(3), pp. 801-820, (1989).

[GERA-91] M. Geradin "The Finite Element Approch to Kinematics and Dymamics of Flexible Multibody Systems", COMET Course, Lingby, Denmark, 27-31 May (1991).

[GERM-86] P. Germain "Mecanique (X)", Edition Marketing, (1986).

[GIBB-879] J.W. Gibbs "On the Fundamentale Formulae of Dynamics, American J. of Mathmatics, Vol. 11, pp. 49-64, (1879).

[GORL-84] B. Gorla & M. Renaud "Modèles des Robots Manipulateurs: Application à leur Commande", Cepadues-Editions, (1984).

[HAML-91] A. Hamlili "Modélisation de l'Etat Dynamique des Robots Manipulateurs et Efficacité du Calcul Formel", Actes du 10^e Congrès Français de Mécanique, Tome (2), 2-6 Sep. (1991).

[HAML-92] A. Hamlili "Application des Systèmes de Calcul Formel pour l'Automatisation de la Modélisation Mathématique des Robots Industriels", Actes du Workshop Calcul formel, INRIA (Sophia-Antipolis), 23-25 Sep. (1992).

[HATO-89] J-P. & M-C. Haton "L'Intelligence Artificielle", PUF, (1989).

[HECK-92] A. Heck "Introduction to Maple", Springer-Verlag, (1992).

[HERV-78] J-M. Hervé "Analyse Structurale des Mécanismes par Groupe des Déplacements", Mechanism and Machine Theory, Vol. 13, pp. 437-450, (1978).

[HERV-82] J-M. Hervé "Intrinsic Formulation of Problems of Geometry and Kinematics of Mechanisms", Mechanism and Machine Theory, Vol. 17, pp. 179-184, (1982).

[HOLL-80] J-M. Hollerbach "An Iterative Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity", IEEE Trans. on Systems, Man and Control, Vol. 10(11), pp. 730–736 (1980).

[HOOK-65] W. Hooker & G. Margulies "The Dynamical Attitude Equation for a N-Bar Satellite", J. of Astronautical Sciences, Vol. 12(4), pp. 123–128, (1965).

[HUET-80] G. Huet "Confluence Reduction: Abstract Proprieties and Applications to Term Rewritings", J. ACM, vol. 27(4), pp. 797-821, (1980).

[HUMP-72] J.E. Humphreys "Introduction to Lie Algebras and Representation Theory", Springer-Verlag, (1972).

[HUN-78a] K.H. Hunt "Spacial Configurations of Robot-Arms via Screw Theory (Part 2)", Robotica, Vol. 5, pp. 17-22, (1978).

[HUN-78b] K.H. Hunt "Kinematic Geometry of Mechanisms", Oxford Engineering Series, Oxford University Press (1978).

[HUST-71] R.L. Huston & C. E. Passerello "On the Dynamics of Human Body Model", J. of Biomechanics, Vol.4, pp. 369-378, (1971).

[HWAN-89] R.S. Hwang & E.J. Haug "Topological Analysis of Multibody Systems Recursive Dynamics Formulations", Mech. Struct. and Mach., vol. 17(2), pp. 239-258, (1989).

[JENK-92] R.D. Jenks & R.S. Sutor "Axiom™: The Scientific Computer System", Nag, Springer-Verlag, (1992).

[JOUA-86] J.P. Jouannaud & E. Kounalis "Automatic Proofs by Induction in Equtional Theories without Constructors, Logic in Computer Science pp. 358-366 (1986).

[KANE-80] T.R. Kane & D.A. Levinson "Formulation of Equations of Motion for Complex Spacecraft", J. of Guidance and Control, Vol. 3, (1980).

[KANE-83] T.R. Kane & D.A. Levinson "The Use of Kane's Dynamical Equations in Robotics", The International Jour. of Robotics Research, Vol. 2(3), pp. 3–21, (1983).

[KANE-85] T.R. Kane & D.A. Levinson "DYNAMICS: Theory and Applications", McGraw-Hill Series in Mechanical Engineering, McGraw-Hill, (1985).

[KAHN-69] M.E. Kahn "The Near Minimum-Time Control of Open-Loop Articulated Kinematic Chains",Standford Artificial Intelligence Laboratory, AI Memo. n° 106, December (1969).

[KAHN-71] M.E. Kahn & B. Roth "The Near Minimum-Time Control of Open-Loop Articulated Kinematic Chains", J. Dynamic Systems Measurment Control, Vol. 93, pp. 167–172, (1971).

[KARG-85] A. Karger & J. Novak "Space Kinematics and Lie Groups", Gordon &Breach Sc. Publishrs, (1985).

[KEAT-87] J.E. Keat "Comment Relationship Between Kane's Equations and the Gibbs–Appel Equations", J. of Guidance and Control Dynamics, Vol. 10, pp. 594–595, (1987).

[KENT-87] R. Kent Dybvig "Lisp: The Scheme Programming Language", Prentice-Hall, New-Jersey, (1987).

[KHAL-78] W. Khalil "Contribution à la Commande Automatique des Manipulateurs avec l'Aide d'un Modèle Mathématique des Mécanismes", Thèse d'Etat, Montpelier, (1978).

[KIRC-90] C. Kirchner, H. Kirchner, M. Rusinowitch "Deduction with Symbolic Constraints", Revue d'Intelligence Artificielle, Vol. 4(3), pp. 9-52, (1990).

[KNUT-70] D.E. Knuth & P.B. Bendix "Simple word Problems in Universal Algebra", Computational Algebra, J. Learch, Pergamon Press, pp 263-297, (1970).

[KŒN-897] G. Kœnig "Leçons de Cinématique", Hermann (1897).

[KOPL-86] J. Koplik and M.C. Leu "Computer Generation of Robot Dynamics Equations and the Related Issues", Journal of Robotic Systems, Vol. 3-3, pp. 301-319 (1986).

[KOUN-90] E. Kounalis & M. Rusinowitch "Mechanizing Inductive Reasoning", Proc. of the AAAI Conference, AAAI Press & MIT Press, pp. 240–245, Boston, (1990).

[KORE-85] Y. Koren "Robotics for Ingineers", McGraw-Hill, (1985).

[LAGR-65] J-L. Lagrange "Mécanique Analytique"[1], Tome 2, Librairie Scientifique et Technique Albert Blanchard, (1965).

[LALE-90] R. Lalement "Logique, Réduction, Résolution", Etudes et Recherches Informatiques, Masson, (1990).

[1]Ouvrage publié après la mort de Lagrange.

[LANG-78] M. Langlois "Sur la Mécanique Analytique du Corps Solide", J. de La Méca., Vol. 17(5), (1978).

[LAUR-76] J.L. Laurière "Alice: un Langage et un Programme pour Résoudre des Problème Combinatoires", Thèse d'Etat, Université Paris VI, (1976).

[LERB-91] J. Lerbet "Mecanismes et Géométrie Différentielle", Actes du 10^e Congrès Français de Mécanique, Tome (4), pp. 205–208, 2-6 Sep. (1991).

[LESC-90] P. Lescanne "On the Recursive Decomposition Ordering with Lexicographical Status and Other Related Orderings", Jour. of Automated Resoning, Vol. 6, pp. 39-49 (1990).

[LIEG-74] A. Liègeois & M. Renaud "Modèle Mathématique des Systèmes Mécaniques Articulés en Vue de la Commande Automatique de leurs Mouvements", CRAS, Vol. 278, Série B, pp. 799–801, (1974).

[LIKI-71] P.W. Likins "Passive and Semi Active Attitude Stabilizations: Flexible Spacecraft", ARGARD, paper 3b. october (1971).

[LIKI-74] P.W. Likins "Dynamic Analysis of Hinge Connected Rigid Bodies with Non-Rigid Appendages", JPL Tech. Report n^o 32–1576, Pasdena, (1974).

[LUH—80] J.Y.S. Luh, M.W. Walker & R.P.C. Paul "On-Line Computational Scheme for Mechanical Manipulators", ASME, Journal of Dynamic Systems, Measurement, and Control, Vol. 112, pp.69-76, (1980).

[LUH—81] J.Y.S. Luh "Automatic Generation of Dynamic Equations for Mechanical Manipulators, J.A.C.C., Charlottesville, (1981).

[MATH-83] The MATHLAB Group Labratory for Computer Science "Macsyma: Reference Manual", MIT, Version 10, Second Printing, Vol.1 and 2, (1983).

[MEGA-84] S. Megahed "Contribution à la Modélisation Géométrique et Dynamique des Robots Manipulateurs ayant une Structure de Chaîne Cinématique Simple ou Complexe", Thèse d'Etat, Toulouse, (1984).

[MERL-87] J-P. Merlet "Contribution à la Formalisation de la Commande par Retour d'Efforts en Robotique: Application à la Commande de Robots Parallèles", Thèse de Doctorat, Paris VI, (1986).

[MEYE-80] B. Meyer et C Baudoin "Méthodes de Programmation", Eyrolles (1980).

[MEYE-12] Meyerson "Identité et Réalité", Librairie Alcan (1912).

[NEW-687] I. Newton "Philosophiae Naturalis Principia Mathematica", Jesus Societatis Ragiae ac Typis Josephi Streater, (1687).

[PAIN-09] P. Painlevé "La Méthode dans les Sciences", Librairie Alcan (1909).

[PAUL-72] R.C.P. Paul "Modelling Trajectory Calculation and Servoing of a Computer Controled Arm", Ph. D. Thesis, Standford University, (1972).

[PAYA-87] S. Payandeh & A.A. Goldenberg "Formulation of the Kinematic Model of a General (6DOF) Manipulator Using Screw Operator", J. of Robotic Syst., Vol, 4(6), pp. 771–797, (1987).

[PITR-90] J. Pitrat "Métaconnaissance: Futur de l'Intelligence Artificielle", Hermès, Paris, (1990).

[ORIN-79] D.E. Orin, R.B. Mc Ghee, M. Vukobratović & G. Hartoch "Kinematique and Kinetic Analysis of Open-Chain Linkages Utilizing Newton–Euler Method", Mathematical Biosciences, n^o 43, pp. 107–130, (1979).

[PAUL-81] R.P. Paul "Robot Manipulators: Mathematics, Programming, and Control", MIT Press, (1981).

[POIN-02] H. Poincaré "La Science et l'Hypothèse", Flammarion, (1902).

[PRAD-89] A.K. Pradeep, P.J. Yoder & R. Mukundan "On the Use of Dual-Matrix Exponentials in Kinematics", The International Journal of Robotics Research, Vol. 8(5) (1989).

[QUEI-90] Ch. Queinnec "A Framework for Data Aggregates", Journées Françaises des Langages Applicatifs, La Rochelle Janvier (1990).

[RAIB-78] M.H. Raibert & B.K.P. Horn "Manipulator Control Using Configuration Space Methode", Industrial Robot, Vol. 5-2, pp.69-73, (1978).

[RAMP-87] R. Rampalli "ADAMS: A Sparse Matrix Approach to Solving Multibody Dynamics Problems", Proc. SDIO/NASA, Workshop

[RAND-84] R.H. Rand "Computer Algebra in Applied Mathmatics: An Introduction to MACSYMA", Research Notes in Mathmatics (94), Pitman Advanced Publishing Program, (1984).

[RENA-80] M. Renaud "Contribution à la Modélisation et à la Commande Dynamique des Robots Manipulateurs", Thèse d'Etat, Toulouse, (1980).

[RENA-85] M. Renaud "A Near Minimum Iterative Analytical Procedure for Obtaining a Robot-Manipulator Dynamic Model", Iuam/IFtomm Symposiumon Dynamics of Multi-Body Systems, Udine, (1985).

[RENA-87] M. Renaud "Quasi-Minimal Computation of the Dynamic Model of a Robot Manipulator Utilizing the Newton-Euler Formalism and the Notion of Augmented Body", Proc. IEEE Conf. on Robotics and Automation, Raleigh, pp. 1667-1682, Mars-April (1987).

[RICO-92] J-M. Rico Martinez & J. Duffy "The Principle of Transference: History, Statement and Proof", Mech. and Mach. Theory, Sended for Publication (1992).

[ROBE-66] R. Roberson & J. Wittenburg "A Dynamical Formalism for an Arbitrary Number of Interconnected Rigid Bodies with Reference to the Problem of Satellite Attitude Control", Proc. 3^{rd} IFAC Congr., Paper 46D, Jun (1966).

[ROSE-87] D.E. Rosenthal "Comment Relationship Between Kane's Equations and the Gibbs–Appel Equations", J. of Guidance and Control Dynamics, Vol. 10, pp. 595–596, (1987).

[RUSI-89] M. Rusinowitch "Démostration Automatique", InterEdition, (1989).

[SANS-85] J.P. Sansonnet "Les machines de l'intelligence artificielle", La Recherche, octobre (1985).

[SERR-65] J-P. Serre "Lie Algebras and Lie Groups", W.A. Benjamain, (1965).

[SHAB-89] A.A. Shabana "Dynamics of Multibody Systems", Wiley, (1989).

[STAN-88] B.J. Stanley "Constrained Motion of a Three-Dimensional Manipulator Over Unknown Constraints: The Robotic Groping Problem", Ph. D. Thesis, Ohio State University, (1988).

[STEE-84] G.L. Steele Jr. "Common LISP : The Language", Digital Press, (1984).

[STEP-76] Y. Stepanenko & M Vukobratović "Dynamics of Articulated Open–Chain Mechanisms", Mathematical Biosciences, n^o 28, pp. 137–170, (1976).

[STRO-86] A. Strohmeier "Fortran 77", Eyrolles, (1986).

[STUR-73] R. Sturges "Teleoperator Arm Design Program", MIT, C.S. Drap. Lab. Cambridge, U.S.A., Repport n° 3.2746, (1973).

[SUG-82a] K. Sugimoto & J. Duffy "Application of Linear Algebra to Screw Systems", Mech. and Mach. Theory vol.17(1), pp. 73-83 (1982).

[SUG-82b] K. Sugimoto, J. Duffy & K.H. Hunt "Special Configurations of Spacial Mechanisms and Robot Arms", Mech. and Mach. Theory vol.17-2, pp.119-132 (1982).

[TOUR-88] P. Tournassoud "Géométrie et Intelligence Artificielle pour les Robots", Hermès, Paris, (1988).

[TSAI-91] F. Tsai & E.J. Haug "Real–Time Multibody System Dynamic Simulation, Part I: Amodified Recursive Formulation and Topological Analysis", Mech. Struct. & Mach., Vol. 19(1), pp. 99-127, (1991).

[UICK-65] J.J. Uicker "On the Dinamic Analysis of Spatial Linkages", Ph.D. Thesis, Northwestern University, (1965).

[UICK-69] J.J. Uicker "Dinamic Behavior of Spatial Linkages", Asme, J. of Ingineering for Industry, Vol.91, pp. 251–258.

[ULLM-65] J. Ullmo "La Pensée Scientifique Moderne", Flamarion, Paris, (1969).

[VALA-91] M. Valàšek "Efficient Implementation of Multibody Formalisms", 8th WGTMM, Prague, Czechoslovakia, (1991).

[VARA-74] V.S. Varadarajan "Lie Groups, Lie Algebras, and their Representations", Prentice-Hall, (1974).

[VELD-82] G.R. Veldkamp "On the Use of Dual Numbers, Vectors, and Matrices in Instantaneous, Spacial Kinematics", Mech. and Mach. Theory, Vol.17-1, pp.73-83, (1982).

[VERE-74] A.F. Vereschagin "Computer Simulation of the Dynamics of Complicated Mechanisms of Robot Manipulaters", Engineering Cibernetics, Vol. 12(6), pp. 65–70, (1974).

[VUKO-85] M. Vukobratović & N. Kircanski "Real Time Dynamics of Manipulation Robots", Springer--Verlag, (1985).

[WATE-79] R.C. Waters "Mechanical Arm Control", AI Laboratory, MIT, AI Memo. n^o 549, October (1979).

[WHIT-04] E.T. Whitteker "A Treatise on The Analytical Dynamics of Particles and Rigid Bodies", Dover Publications, (1904).

[WITT-75] J. Wittenburg & L. Lilov "Relative Equilibrium Positions and their Stability for a Multi-body Satellite in a Circular Orbit", Ingenieer Archiv, Vol. 44 pp. 269–279, (1975).

[WITT-77] J. Wittenburg "Dynamics of Systems of Rigide Bodies", B.G. Teubner, (1977).

[WITT-85] J. Wittenburg & U. Woltz "MESA VERDE: a Symbolic Program for Non-linear Articulated Rigid-Body Dynamics. Proc. ACME Design Eng. Div. Conf. and Exibit on Mechanical Vibration and Noise, Cincinati, (1985).

[YAN-64a] A.T. Yang "Application of Dual Number Quaternion Algebra to the Analysis of Spatial Mechanisms", ASME J. for Indus. 31(2), pp. 300-308 (1964).

[YAN-64b] A.T. Yang and F. Freudenstein "Application of Dual Number Quaternion Algebra to the Analysis of Spacial Mechanismes", J. Appl. Mech., vol. 86, pp. 300-308 (1964).

[YOSH-84] T. Yoshikawa "Manipulatibility of Robotics Mechanisme", Proc. 2nd Int. Symp. of Robotics Research, Kyoto, pp. 91-98, (1984).

Résumé

Dans cette thèse nous apportons deux contributions importantes par l'outil de l'abstraction mathématique:

- *La première contribution concerne la mécanique et plus précisément la modélisation dynamique des systèmes articulés. L'abstraction mathématique par la théorie des groupes et algèbres de Lie coordonnée avec un usage judicieux de la notion des nombres duaux permettent d'élaborer un langage très commode où les modèles géométriques et dynamiques des systèmes mécaniques poly-articulés s'expriment sous une forme syntaxique relativement simple (malgré la complexité du système). De nouvelles méthodes pour la description des configurations des systèmes multicorps et un algorithme récurrent original (et très efficace) sont alors développés grâce à ce langage.*

- *La seconde contribution concerne le domaine informatique en calcul formel. Elle est basée sur le typage algébrique, les techniques de réécriture et la génération automatique des codes (programmation assistée par ordinateur). Les problèmes soulevés nécessitent de nouvelles architectures de systèmes de calcul formel. Dans cet ordre d'idées, un prototype de système de calcul formel (SURVEYOR) basé sur la réécriture typée et une extension (MEDUSA MF77) du systèmes Maple ont été réalisés. Un outil informatique pour la génération automatique des codes Fortran et Maple des schémas de calcul optimisés relatifs à notre formulation dynamique est développé à l'aide du système MEDUSA MF77.*

Plusieurs applications en calcul symbolique et en robotique sont, par ailleurs, présentées en annexes sous forme de réalisations informatiques des aspects théoriques traités.

Mots clés:

Cinématique, Dynamique, Chaînes Cinématique ouvertes, Géométrie Différentielle, Groupes et Algèbres de Lie, Intelligence Artificielle, Programmation Orientée Objets, Prototypage de Systèmes de Calcul Formel, Génération Automatique de Codes.